рНАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

«До захисту допущено»

Завідувач кафедри
_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)
“___” червня 2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки **6.050102 «Комп'ютерна інженерія»**
на тему: Веб-додаток для подання та перевірки мінімальності комбінаційних схем на базі двовходових логічних елементів

Виконав: студент IV курсу, групи КВ-51
(шифр групи)

Марченко Олександр Борисович
(прізвище, ім’я, по батькові) (підпис)

Керівник доц.каф.СПСКС, к.т.н. Потапова К.Р.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М.
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____
_____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050102 «Комп’ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)
«___» червня 2019 р.

ЗАВДАННЯ
на дипломний проект студента
Марченко Олександра Борисовича
(прізвище, ім’я, по батькові)

1. Тема проекту

Веб-додаток для подання та перевірки мінімальності комбінаційних схем на базі двовходових логічних елементів, керівник проекту Потапова Катерина Романівна к.т.н. доцент, затверджені наказом по університету від «22» травня 2019 р. №1330-С

2. Термін подання студентом проекту «___» червня 2019 р.

3. Вихідні дані до проекту див. Технічне завдання

4. Зміст пояснювальної записки

- Аналіз існуючих рішень та обґрунтування теми дипломного проекту
- Методи та технології
- Веб-додаток

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

- База даних. Схема структурна
- Компіляція схем. Схема алгоритму
- Мінімізація функції . Схема алгоритму
- Міжмодульна взаємодія. Схема структурна

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М. доцент		

7. Дата видачі завдання «___» _____2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1	Вивчення літератури за тематикою проекту	17.11.2018	
2	Розроблення та узгодження технічного завдання	28.11.2018	
3	Аналіз існуючих рішень	15.12.2018	
4	Підготовка матеріалів першого розділу дипломного проекту	30.12.2018	
5	Розроблення програмного забезпечення	03.02.2019	
6	Відлагодження програмного продукту	10.02.2019	
7	Підготовка матеріалів другого розділу дипломного проекту	20.02.2019	
8	Підготовка матеріалів третього розділу дипломного проекту	30.03.2019	
9	Підготовка графічної частини дипломного проекту	19.05.2019	
10	Оформлення документації дипломного проекту	26.05.2019	

Студент _____ Марченко О.Б.
(підпис) (ініціали, прізвище)

Керівник проекту _____ Потапова К.Р.
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку (52 сторінки, 17 рисунків)

Об’єкт розробки – створення веб-додатку для представлення та перевірки мінімальності комбінаційних схем на базі двовходових логічних елементів.

Веб-додаток дозволяє будувати комбінаційні схеми та перевіряти їх на мінімальність. Для рішення поставленого завдання було створено два програмні модулі: модуль побудови логічних схем та модуль аналізу логічних схем. Модуль аналізу реалізує алгоритм мінімізації булевих функцій за допомогою метода Квайна-Мак-Класкі. Результатом роботи алгоритму є висновок про мінімальність комбінаційної схеми.

Додаток побудований на платформі ASP.NET Core. Обрана мова програмування: C# для серверної частини та JavaScript (JQuery) для взаємодії с користувачем. Звернення до бази данних (MSSQL) відбуваються за допомогою EntityFramework, за візуалізацію схем відповідає бібліотека Vis.js, взаємодія з сервером відбувається за допомогою API контроллерів та Razor Pages.

В ході розробки:

– проведено аналіз існуючих рішень для створення, подання та аналізу комбінаційних схем;

– сформульовані вимоги до веб-додатку для представлення та перевірки мінімальності комбінаційних схем;

– розроблено корстувацький інтерфейс для створення комбінаційних схем;

– розроблено модуль подання комбінаційних схем;

– розроблено веб-сервіси для знаходження таблиці істиності та перевірки на мінімальність комбінаційних схем.

Ключові слова:

АНАЛІЗ КОМБІНАЦІЙНИХ СХЕМ, МІНІМІЗАЦІЯ КОМБІНАЦІЙНИХ СХЕМ, ПОДАННЯ КОМБІНАЦІЙНИХ СХЕМ, МЕТОД КВАЙНА-МАК-КЛАСКІ, C#, ASP.NET Core, Entity Framework

ANOTATION

Qualification work includes an explanatory note (52 pages, 17 images)

The object of development is the creation of a computer web application for representing and checking the minimality of combinational circuits based on two inputs of logic elements.

The computer system allows you to build combinational circuits; check circuits for minimality. To solve the pre-diploma practice, two software modules were created: a module for constructing logic circuits and a module for analyzing logic circuits. The analysis module implements an algorithm for minimizing Boolean functions using the Quine method. The result of the algorithm is information about the minimality of the combinational circuit.

The application is built on the ASP.NET Core platform. Selected programming language: C # for server-side and JavaScript (jQuery) for user interaction. Database calls (MSSQL) are performed using EntityFramework, Vis.js library is responsible for schema visualization, server interaction is performed using controllers API and Razor Pages.

During development:

- analysis of existing solutions for the construction and analysis of combinational circuits;
- formulated the requirements for the web application to represent and verify the minimality of combinational circuits;
- developed a user interface for creating combinational circuits;
- developed a module for viewing combinational circuits;
- developed web services to find the truth table and check for minimal combinational circuits.

Keywords:

ANALYSIS COMBINATIONAL CIRCUITS, MINIMIZATION COMBINATIONAL CIRCUITS, MODELING COMBINATIONAL CIRCUITS, METHOD QUINE-MC-CLASKEY, C#, ASP.NET Core, Entity Framework, MSSQ

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ. 045440.002 ТЗ	Веб-додаток для подання	4		
			та перевірки мінімальності			
			комбінаційних схем на базі			
			двовходових логічних			
			елементів			
			Технічне завдання			
	А4	ІАЛЦ.045440.003 ТП	Веб-додаток для подання	1		
			та перевірки мінімальності			
			комбінаційних схем на базі			
			двовходових логічних			
			елементів			
			Відомість технічного			
			проекту			
	А4	ІАЛЦ.045440.004 ПЗ	Веб-додаток для подання	52		
			та перевірки мінімальності			
			комбінаційних схем на базі			
			двовходових логічних			
			елементів			
			Пояснювальна записка			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.001 ОА		
Розробив		Марченко О.Б.			Веб-додаток для подання та перевірки мінімальності комбінаційних схем на базі двовходових логічних елементів. ***Опис альбому***	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Потапова К.Р.					
Консульт.						НТУУ «КПІ ім. І. Сікорського» Кафедра СПіСКС Група КВ-51	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.005 Д1	База даних.	1		
			Схема структурна			
	А4	ІАЛЦ.045440.006 Д2	Компіляція схем.	1		
			Схема алгоритму			
	А4	ІАЛЦ.045440.007 Д3	Мінімізація функції.	1		
			Схема алгоритму			
	А4	ІАЛЦ.045440.008 Д4	Міжмодульна взаємодія.	1		
			Схема структурна			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.001 ОА	Арк. 2

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Веб-додаток для подання та перевірки мінімальності комбінаційних схем на базі двовходових логічних елементів».

Галузь застосування: дослідження способів оптимізації роботи систем обчислення.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення веб-додатку, що дозволяв би моделювати комбінаційні схеми і перевіряти їх мінімальність.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- сумісність з будь якою операційною системою де встановлено бразуер;

- можливість створення комбінаційних схем;
- можливість візуалізації комбінаційних схем;
- можливість редагування комбінаційних схем;
- можливість перегляду таблиці істиности;
- наявність інформації про мінімальність схеми;
- зберігання схем.

5.2. Вимоги до програмного та апаратного забезпечення користувача

- Процесор: MediaTek, Snapdragon, Kirin, Intel, AMD;
- Оперативна пам’ять: 2 Гб;
- Наявність доступу до мережі Internet;
- Операційна система Windows Phone, Android, iOS, Windows, Linux, Mac OS.

5.3. Вимоги до програмного та апаратного забезпечення серверу

- Процесор: Intel, AMD;
- Оперативна пам’ять: 4 Гб;
- Наявність доступу до мережі Internet;
- Операційна система Windows, Linux.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Вивчення літератури за тематикою проекту	15.04.2019
2.	Розроблення та узгодження технічного завдання	30.04.2019
3.	Аналіз існуючих рішень	05.05.2019
4.	Підготовка матеріалів першого розділу дипломного проекту	10.05.2019
5.	Підготовка матеріалів другого розділу дипломного проекту	18.05.2019
6.	Підготовка графічної частини дипломного проекту	20.05.2019
7.	Оформлення документації дипломного проекту	25.05.2019
8.	Попередній огляд матеріалів диплому на кафедрі	28.05.2019

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045440.004 ПЗ	Веб-додаток для подання	52		
			та перевірки мінімальності			
			комбінаційних схем на базі			
			двовходових логічних			
			елементів			
			Пояснювальна записка			
	А4	ІАЛЦ.045440.005 Д1	База даних.	1		
			Схема структурна			
	А4	ІАЛЦ.045440.006 Д2	Компіляція схем.	1		
			Схема алгоритму			
	А4	ІАЛЦ.045440.007 Д3	Мінімізація функції.	1		
			Схема алгоритму			
	А4	ІАЛЦ.045440.008 Д4	Міжмодульна взаємодія.	1		
			Схема структурна			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.045440.003 ТП		
Розробив		Марченко О.Б.			Веб-додаток для подання та перевірки мінімальності комбінаційних схем на базі двовходових логічних елементів.	Літ.	Аркуш 1 / Аркушів 1
Перевірила		Потапова К.Р.					
Консульт.						НТУУ «КПІ ім. І. Сікорського» Кафедра СПіСКС Група КВ-51	
Н. контроль		Клятченко Я.М.					
Зав. каф.		Тарасенко В.П.			***Відомість технічного проекту***		

ЗМІСТ

					ІАЛЦ.045440.004ПЗ		
Зм	***Лист***	***№ докум.***	***Підп.***	***Дата***			
Розроб.		Марченко О.Б.			Веб-додаток для подання та перевірки мінімальності комбінаційних схем на базі двовходових логічних елементів. ***Пояснювальна записка***	***Літ.***	***Лист*** 1 / ***Листів*** 52
Перев.		Потапова К.Р.					
						НТУУ «КПІ ім. І. Сікорського»,ФПМ,КВ-51	
Н. контр.		Клятченко Я.М.					
Затв.		Тарасенко В.П.					

ДОДАТКИ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

.NET Core – програмна платформа.

MSSQL– система управління реляційними базами данних.

C# – мова програмування високого рівня.

ПЗ – програмне забезпечення.

ВСТУП

Метою дипломного проекту є розробка веб-додатку, що дозволяє задавати комбінаційні схеми, що складаються з двовходових елементів та перевіряти їх на мінімальність. Зараз не існує програмного комплексу у вільному доступі, який би виконував обидві ці функції. Існує досить велика кількість різноманітних засобів для побудови логічних схем, але більшість з них обмежена за багатьма параметрами.

У електроніці логічний елемент є ідеалізованим або фізичним пристроєм, що реалізує булеву функцію; тобто виконує логічну операцію на одному або декількох двійкових входах і має один двійковий вихід. Для дослідження комбінаційних схем зазвичай посилаються на ідеальні логічні елементи, які мають нульовий час реагування на вхідний сигнал та необмежену кількість елементів підключених до виходу.

Логічні елементи в основному реалізуються за допомогою діодів або транзисторів, що діють як електронні комутатори, крім того вони можуть бути побудовані з використанням вакуумних трубок, електромагнітних реле (релейна логіка), флюїдної логіки, пневматичної логіки, оптики, молекул або навіть механічних елементів. Завдяки підсиленю логічні елементи можуть бути каскадними так само, як булеві функції можуть бути багаторівневими, дозволяючи побудувати фізичну модель всієї булевої логіки, а отже, і всі алгоритми, які можна описати за допомогою булевої логіки.

Логічні схеми включають такі пристрої, як мультиплексори, регістри, арифметичні логічні блоки (АЛП), пам'ять комп'ютера, і навіть повні мікропроцесори, які можуть містити більше 100 мільйонів логічних елементів. У сучасній практиці більшість елементів виготовляються з польових транзисторів, зокрема, металооксидних напівпровідникових польових транзисторів.

Деякі логічні елементи І-АБО-НЕ та АБО-І-НЕ часто використовуються в схемотехніці, оскільки їх побудова з використанням польвоих транзисторів є більш простим і ефективним способом, ніж сума окремих елементів, саме тому при виконані дипломного проекту було закладено можливість створення логічних елементів з будь-якою логікою (шляхом введеня таблиць істиності цих елементів).

Розроблений програмний продукт вирішує поставлені задачі. Веб-додаток надає користувачу можливості побудови різноманітних логічних схем на будь-якій платформі, без глибоких знань принципів проектування. Для більшої гнучкості роботи - було прибрано обмеження на використання логічних елементів, тому при налуштувані клієнт може створити власні елементи шляхом занесення їх таблиць істиності до інтерфейсу.

Функціонал подання логічних схем надає можливості для візуального аналізу, використовуючи когнітивні можливості людини, і дозволяє виявити помилки побудови, а також відкоректувати положення елементів та їх портів. Якщо графічного відображення недостатньо для виявлення проблем можна запустити компіляцію схеми, виходом якої буде таблиця істиності. Для більш детального розгляднення поведінки окремих елемінтів достатньо змінити режим відображення таблиці на розширений, в якому відображаються значення на всіх лініях комбінаційної схеми.

В дипломній роботі було автоматизовано мінімізацію будь-яких комбінаційних схем, що дозволяє виявити надлишковость ще на етапі проектування. Для досягнення максимальних результатів швидкодії, адже як відомо процес мінімізації потребує досить великої кількості ітерацій, були проаналізовані аналітичні та табличні методи мінімізації, серед яких було обрано найшвидше рішення.

Проект легко розгортається на будь-якій операційній системі, оскільки для його розробки було використано сучасну платформу .Net Core. Користувацький інтерфейс повністю адаптовано під усі сучасні пристрої від

смартфонів до насітльних комп’ютерів. Це дозволяє використовувати розроблене програмне забезпечення будь-де і без витрат на облаштування робочого місця інженера комп’ютером.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

1.1. Проблема, яку вирішує продукт

Найперше, що хотілось би зазначити – це те, що існуючі програмні продукти зазвичай не дають змоги створення власних логічних елементів і, відповідно, обмежені у можливості користувацької модифікації елементів. Крім того можливості аналізу залишають бажати кращого. Якщо з побудовую таблиць істинности деяки продукти справляються, то можливостей аналізу булевих функцій комбінаційних схем, мінімізації, зміни логіки роботи елементів, пошуку помилок та інших вони, нажаль, не надають.

1.2. Аналоги. Переваги та недоліки

Розглянемо одне з існуючих рішень – CircuitVerse. CircuitVerse - продукт, розроблений студентами університету Бангалора. Мета проєкту - забезпечити платформу, де схеми можуть бути спроектовані та імітовані за допомогою графічного інтерфейсу користувача. Хоча користувачі можуть розробляти повну реалізацію процесора в симуляторі, програмне забезпечення призначене в основному для освітніх цілей.

Проектування схеми відбувається швидко і легко, за це відповідає сучасний і інтуїтивно зрозумілий інтерфейс користувача. Для створення схеми використовуються такі елементи взаємодії з користувачем: перетягування і відпускання, копіювання / вставка, масштабуванням.

Слід зазначити, що CircuitVerse підтримує багатоканальні дроти, це дає широкий простір для моделювання різноманітних схем великої складності. Для зміни бітової ширини елементів схеми необхідно просто виділіти елемент схеми і змініти значення бітової ширини в меню властивостей.

Наявність можливості створювати підсхеми і використовувати їх повторно дозволяє легше і більш структуровано проектувати. Розробники обіцяють, що у майбутніх версіях дозволять імпортувати підсхеми з зовнішніх джерел. Цю особливість я використаю у своїй дипломній роботі, але у вигляді можливості створення логічних елментів необмеженої кількості входів та виходів та збереження їх таблиць істиності.

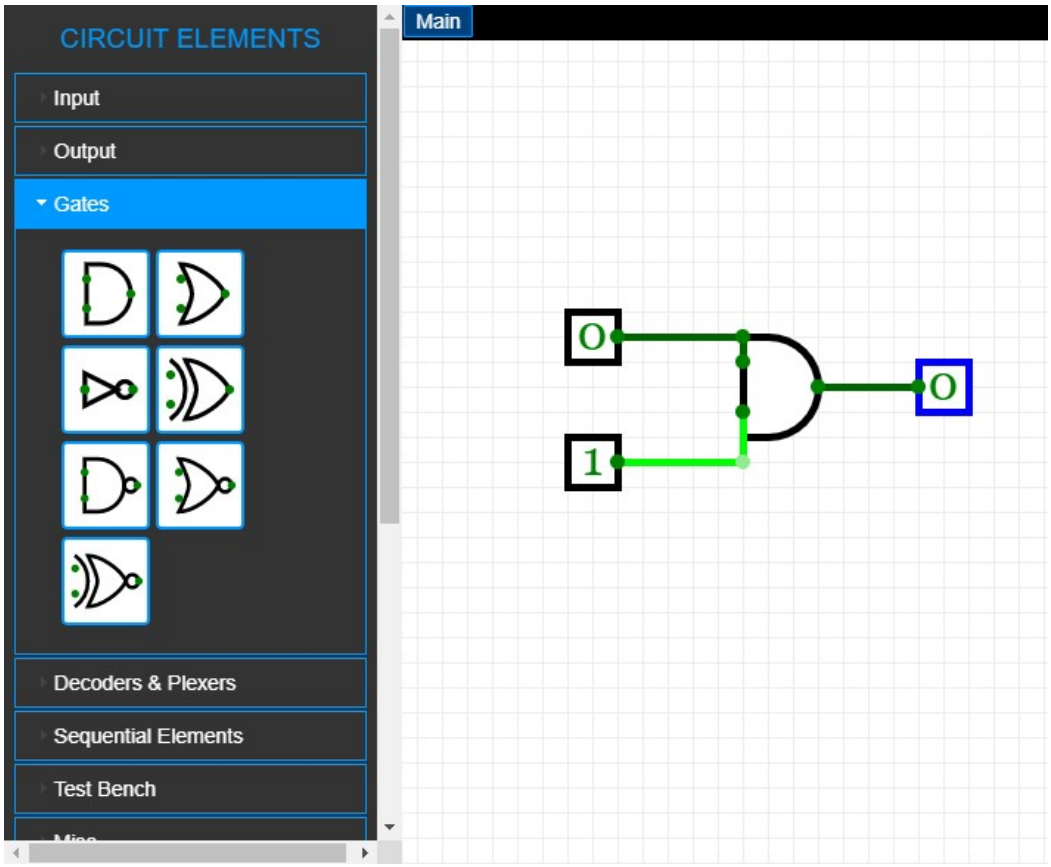


Рис. 1

Як ми бачимо на рис.1 інтерфейс має зручну підствітку ділянок схеми, на яких знаходиться логічна одиниця.

Також є дуже цікава функція «Група» - це функція, створена для академічних цілей. Наставник може створити групу і запросити різних членів приєднатися до неї. Пізніше наставник може створювати різні завдання, які можуть використовуватися для оцінки прогресу членів групи. Особливості цієї функції:

- Наставник встановлює опис або завдання для завдання.
- Потім наставник встановлює термін, після якого завдання членів автоматично надсилаються.
- Члени можуть продовжувати свою роботу з відповідними завданнями навіть після закінчення кінцевого терміну, але ці зміни не будуть відображені в призначенні, яке було надано наставнику.
- Ментор може повторно відкрити або змінити призначення в будь-який конкретний час.

Серед менш значних переваг:

- CircuitVerse може експортувати зображення з високою роздільною здатністю в різних форматах, включаючи SVG.
- Автоматично генерувати схему на основі даних таблиці істиності. Це допомогає для створення складних логічних схем і може бути легко зроблено в підсхему.
- Оскільки CircuitVerse вбудована в HTML5, для кожного проекту може бути згенерований iFrame, що дозволяє користувачеві впроваджувати його практично в будь-який сайт.

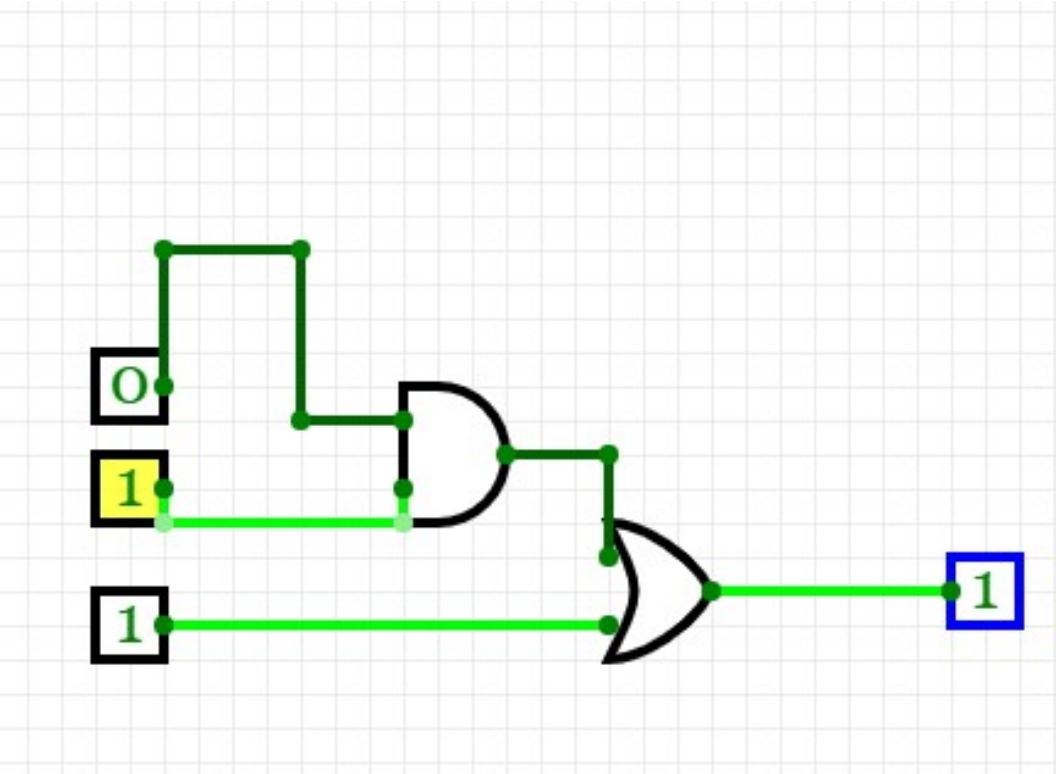


Рис. 2

Оскільки система не є комерційною і підтримується виключно з ініціативи розробників вона не позбавлена недоліків, серед самих значних можна виділити:

- Відсутність можливості створення таблиці істиності на основі побудованої схеми
- Відсутність можливостей аналізу схеми, таких як перевірка на мінімальність, пошук помилок та інше.
- Неможливість інтеграції стороніх модулів у систему.
- Обмеженість системи конкретною задачею. Оскільки не можна змінювати назви елементів системи досить важко моделювати процеси поза двійковою логікою.
- З рис.1 та рис.2 видно, що елементи позначені не за Українськими стандартами.

- Також не має можливості пересувати зв’язки між елементами, а автогенеровані досить незручно читати (приклад рис.2).
- Немає можливості задати схему без задання значень на входах.
- Немає можливості відмінити останні зміни, а при ручному видалені і редагувані схеми інтерфейс часто поводить себе некорректно (приклад рис. 2).

Отже базуючись на зазначеному вище CircuitVerse підходить для побудови простих логічних схем, в першу чергу програма орієнтована на емуляцію роботи при заданих значеннях і допомагає виконувати ручний аналіз схем трохи наглядніше. Проте кількість шаблонізованих елементів може бути у нагоді при моделювані процесорної логіки. Дана програма чудово підійде в навчальних цілях за рахунок своїх функцій, але для більш глобальних задач її не вистачає функціоналу та стабільності роботи.

Наступним рішенням, яке я хотів би розглянути є logic.ly. Проєктування схеми тут відбувається за допомогою схожого на CircuitVerse користувальницького інтерфейсу: перетягування і відпускання, копіювання / вставка, масштабування.

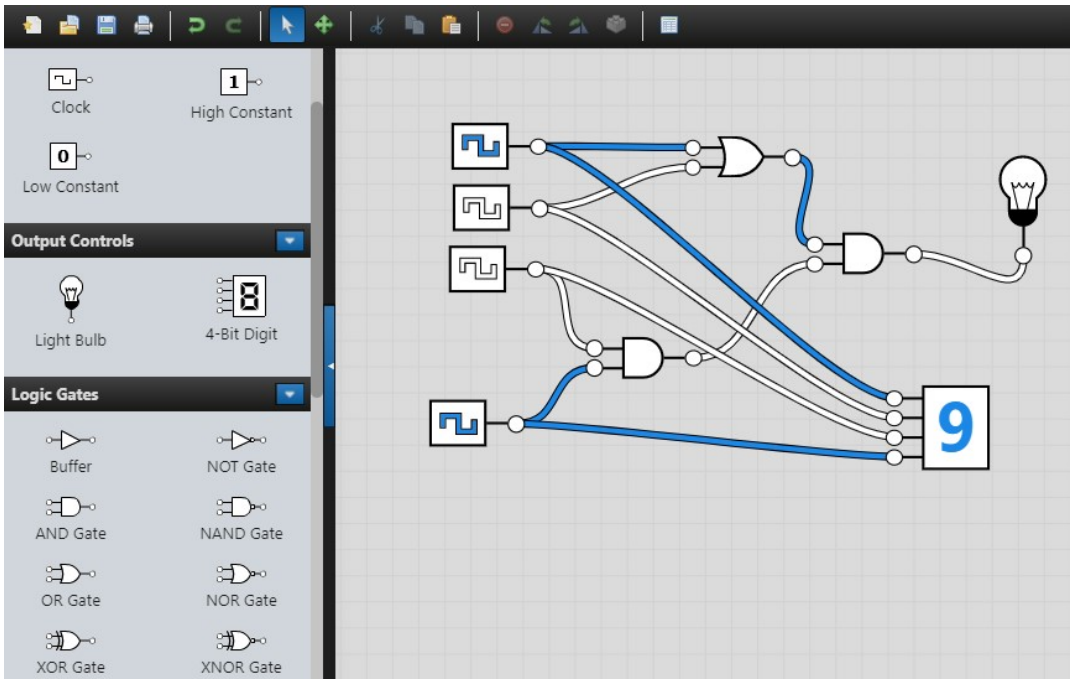


Рис. 3

Значною перевагою порівняно з минулою програмою є можливість контролювати налагодження. У системі шаблонізовано елементи, що дозволяють імітувати роботу схеми, зупиняти і запускати імітацію та спостерігати, як сигнал поширюється в колі, крок за кроком. Система не є кроссплатформеною, але підтримується на Windows, так і MacOS.

Програмний комплекс орієнтований на навчальні заклади, э платним і досить дорогим. Також до переваг можна віднести:

- Наглядність роботи створеної схеми (рис. 3).
- Можливість побудови таблиці істиності для окремих ділянок схеми.
- Можливість відміняти та повторювати останні дії.

Недоліки системи:

- Завищена вартість, як для освітньої програми.
- Відсутність підтримки мобільних засобів (планшетів, смартфонів) та операційної системи Linux.

- Хоча і є можливість емулювати роботу схеми, відстежувати стан схеми досить незручно.
- Відсутність можливостей автоматичного аналізу схеми.
- Закритість коду, відповідно неможливо вдосконалити програму силами спільноти, або доробити модулі для власного використання.
- Як і CircuitVerse – Logic.ly не підтримує імена елементів схеми, що робить неможливим використання схеми для демонстрації логіки на окремих її ділянках.
- Відсутність Українських умовних позначень елементів.
- Інколи при роботі з інтерфесом елементи стають неактивними (їх неможна виділити, видалити і доводиться створювати схему спочатку).

Базуючись на зазначеному вище можна сказати, що Logic.Ly досить симпатичний продукт, що підійде для презентацій роботи тих чи інших комбінаційних схем, але досить обмежений і дуже дорогий.

Далі я хотів би розглянути рішення creately. Це рішення в першу чергу цікаве за можливості побудови різного роду діаграм, хоча воно не так сильно орієнтовано на побудову комбінаційних схем, проте має значну кількість інтеграцій зі стороніми сервісами, такими як JIRA, GSuite, Confluence та інші.

Цей додаток дозволяє створювати прості та складні логічні елементи - від анотованих електричних схем до цифрових та аналогових логічних проектів.

Генератор комбінаційних схем Creately пропонує широкий спектр унікальних функцій для швидкого малювання діаграм з логічних елементів (рис. 5). Додаваня елементів на схему спрощено, тут немає необхідності як в минулих прикладах перетягування та вручну підключати елементи, завдяки функції створення та підключення це відбувається одним клацанням миші.

Існує цікава особливість у вигляді розумних форми та роз'ємів, що автоматично налаштовуються відповідно до схеми, тому тут не доводиться вручну переставляти елементи після мінімальних змін в схемі. Це само по собі заощаджує багато часу при складанні великих логічних схем. Creately окрім зручного інтерфейсу створення схем ще й має функцію інтелектуальних форм, що зберігає дані та налаштовує взаємозв’язок з іншими фігурами на діаграмі і автоматично обчислює вихідні значення схеми. Це відмінно підходить для навчання.

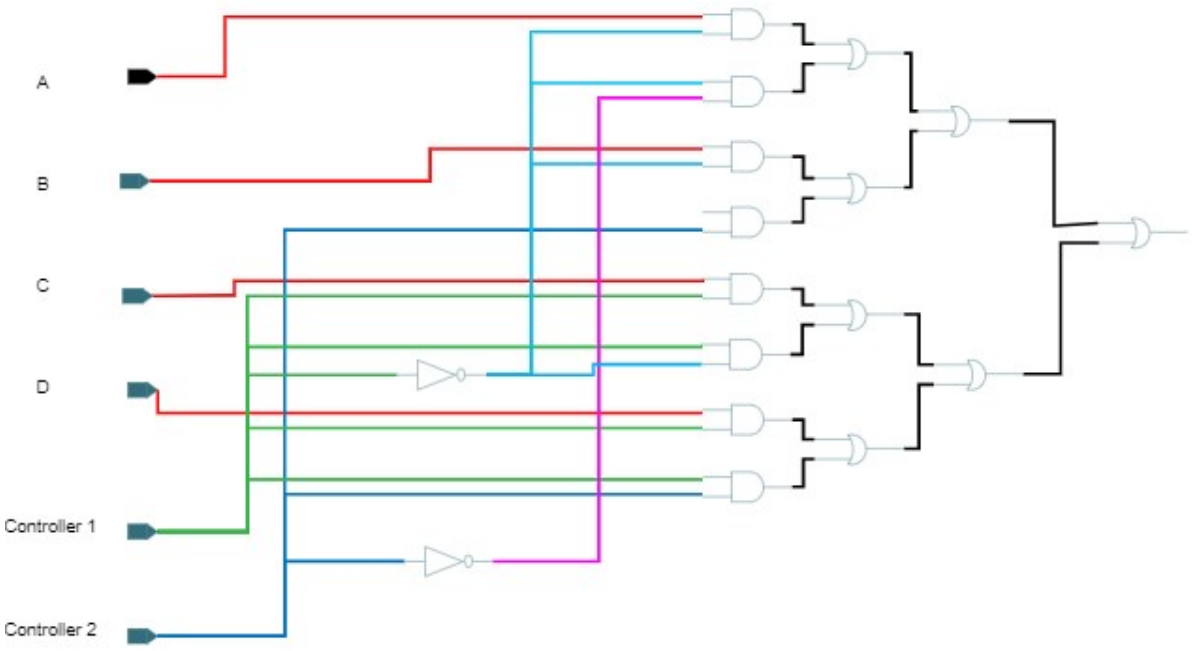


Рис. 4 побудова мультиплексора в CreatLy

Creatly надає декілька способів представлення та експорту логічних схем. Можна експортувати діаграми у форматі PDF або SVG.

Експорт SVG надзвичайно корисний, оскільки довзволяє продемонструвати результати своєї роботи будь кому. SVG підтримується всіма сучасними браузерами та багатьма іншими інструментами діаграм.

Проте при створені великиї схем, таких як зображено на рисунку 4, виникають проблеми з відображенням їх відображенням, існує додаток Creately Viewer, що спеціально розроблений, щоб допомогти користувачам вставляти будь-яку велику діаграму в невеликий простір. Збільшення масштабу, зменшення масштабу, перетягування та виконання набагато більше. Ця особливість є ключовою для даного проєкту, оскільки великі схеми не можна розмістити на маленьком екрані без втрати читаємості. Нажаль Creatly не підтримується мобільними телефонами, адже простір мобільних телефонів дуже обмежений, цей недолік я планую виправити, про що буде сказано нижче.

Створення логічних схем є складним процесом, і іноді можуть виникнути проблеми с тим, щоб зрузуміти де було допущено помилку. Creatly надає можливості співпраці в реальному часі. Я вважаю, що це дуже корисна функція, що необхідна кожному сучасному веб-додатку, тому цей функціонал також реалізован в цій роботі. Проте Creatly ще дає можливість бачити коментарі до схеми, зроблені кимось з команди в реальному часі, щоб користувач міг виправити будь-яку проблему одразу. Я вважаю, що цей зайва функція адже займає частину і без того ціного інтерфейсу і постіно опитує серевер на предмет нових повідомлень, що в свою чергу витрачає ресурси клієнта в тому числі. Для засобів зв’язку з команою слід використовувати стороні програми, що спецільно розроблені для цього як наприклад Slack.

Кожна зміна зберігається в історії ревізій, це дозволяє легко проаналізувати зміни та відмінити їх у разі потреби.

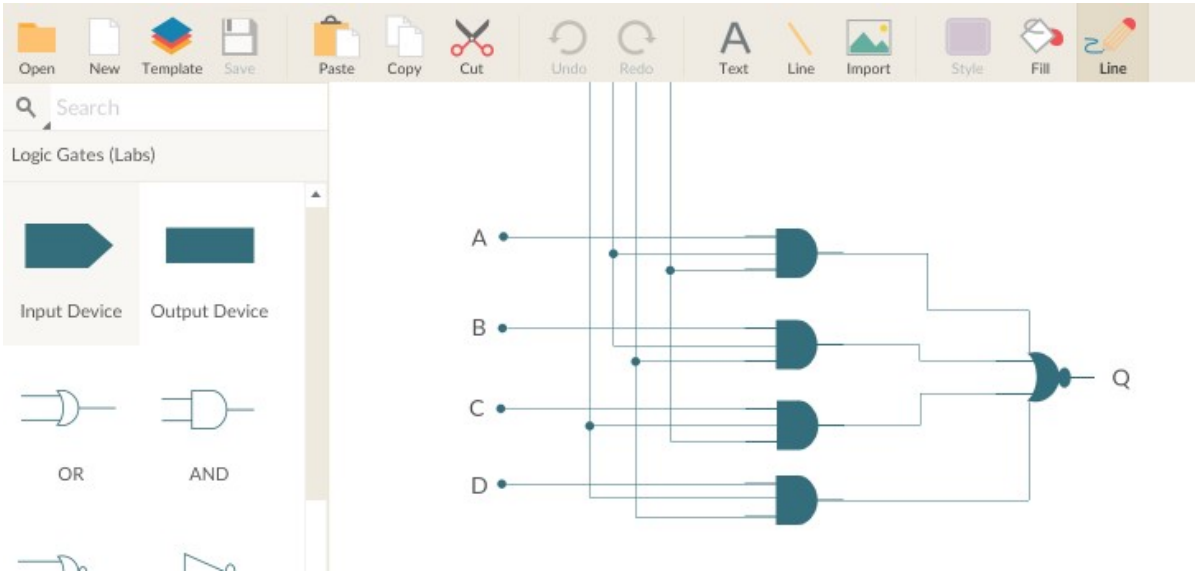


Рис. 5 Зовнішній вигляд інтерфейсу CreatLy

Найбільшими недоліками системи є відсутність аналітичних можливостей над отриманими схемами та потреба у запуску FLASH в браузері. Як відомо – ця технологія є застарілою в наш час і містить велику кількість вразливостей, якими можуть скористатись зловмисники. Крім того сервіс потребує авторизації та дуже повільно працює.

Не менш цікавим рішенням є рішеня від світового дитрибьютора електрокомпонентів digikey під назвою SchemeIt.

SchemeIt – це онлайн додаток для побудови схем і діаграм, що дозволяє будь-якому користувачеві розробляти і ділитися електронними схемами. Інструмент включає в себе всеосяжну електронну бібліотеку елементів і інтегрований каталог компонентів Digi-Key, що дозволяє використовувати широкий спектр електроних схем. Крім того, передбачено вбудований менеджер з обліку матеріалів для відстеження деталей, що використовуються в проекті (рис. 6). Після того, як схематичне креслення завершено, користувачі можуть експортувати його до файлу у вигляді зображеня або поділитися ним по електронній пошті з іншими. Схема-вона працює у всіх основних веб-браузерах без необхідності використання будь-яких плагінів, проте не підтримується мобільними пристроями. Для використаня системи потрібно

лише авторизуватися, для доступу до можливостей збереження та експорту схем. До переваг можна віднести:

- Можливість побудови діаграм на рівні блоків
- Доступ до більш ніж 4 мільйонів компонентів за допомогою інтеграції з каталогом Digi-Key
- Функціонал зберігання проектів як приватних або публічних, з можливістю відправки посилання на створені схеми, або вбудовування їх в веб-сторінки, блоги або електронні листи.
- Є підтримка імпортування за стандартом BOM, що дозволяє створювати схеми на основі даних обліку існуючих матеріалів.
- Експорт у файли PDF або PNG.

Також за рахунок комерційної зацікавленості компанії є пряма інтеграція з службою технічної підтримки Digi-Key для допомоги у виборі компонентів, зо може стати у нагоді при проектувані.

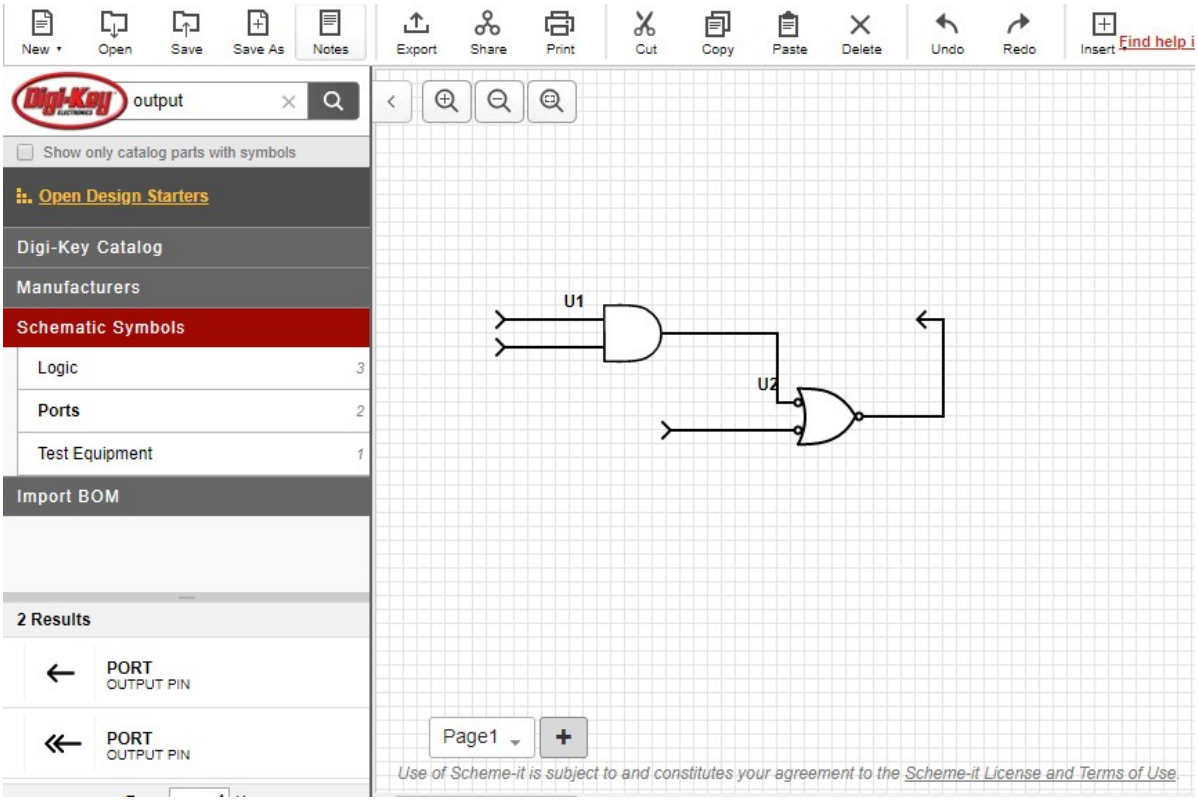


Рис. 6 Інтерфейс SchemeIt

Головною переваго. цього додатку над попередніми можна назвати його можливості у побудові будь-яких пристроїв через величезну бібліотеку компонентів. Цей додаток відміно підійде інженерам та компаніям, що спеціалізуються на виготовлені радіо-компонентів. Як і в минулих додатках головним недоліком є повна відсутність будь-яких аналітичних можливостей.

Ще один цікавий додаток – visual paradigm (рис. 7). Це рішення чудово підійде для малювання логічних схем? Інструмент Visual Paradigm має зручний редактор діаграм, що дозволяє швидко малювати логічні схеми. Дане програмне забезпечення має всі типи логічних елементів, необхідні для розробки будь-якої логічної моделі.

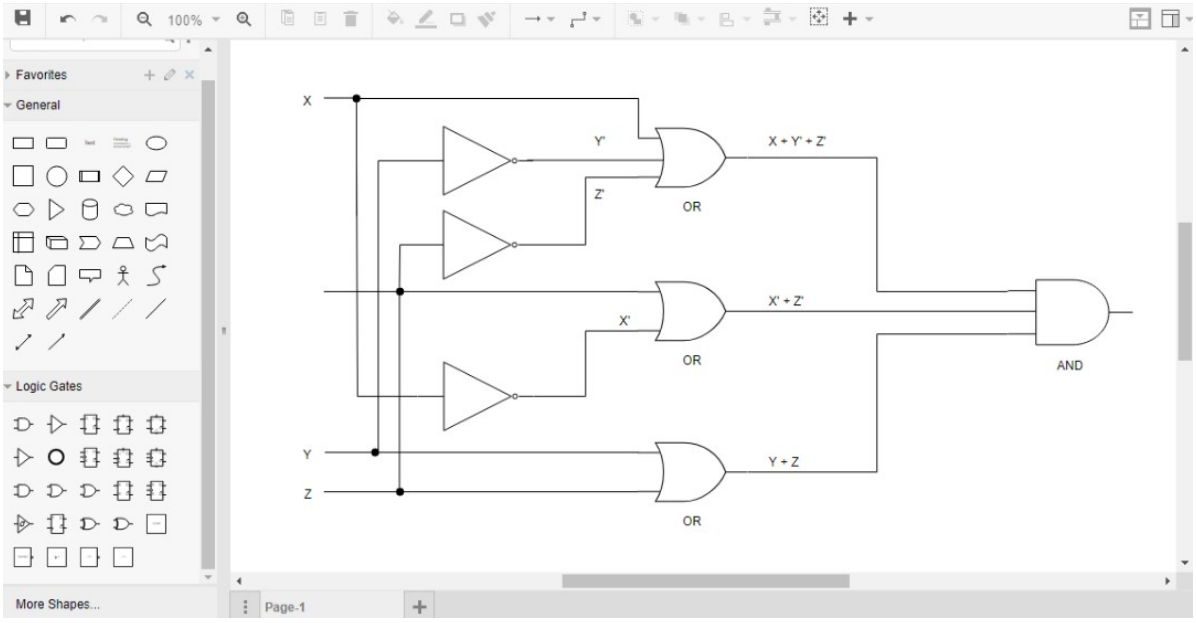


Рис. 7 Інтерфейс Visual Paradigm

Як і більшість інших сервісів створення схем тут відбувається за допомогою перетягування та поєднання роз'ємів. Проте тут присутня дуже цікава особливість, така як авто-вирівнювання, що допомогає точно розташувати елементи під час перетягування.

Замість того, щоб копіювати і вставляти статичні зображення, даний додаток надаэ можлливість вбудовувати інтерактивні логічні схеми в додатки MicroSoft. Підтримуються Word, PowerPoint, OneNote і Outlook.

Є можливість імпорту з Visio. Можна використовувати трафарети (.vssx, .vsdx) в рамках додатку за допомогою функції імпорту. Після імпорту схему можна відредагувати.

Поділіться своєю роботою з іншими можна, друкуючи або експортуючи діаграми у форматі зображень (PNG, JPG, SVG, GIF) або PDF.

У системі є досить велика кількість заготовлених шаблонів. Можна опублікувати власні схеми в загальний доступ, щоб інші могли ними скористатися.

Підводячи висновки щодо даного додатку слід зазначити, що його функціональність для побудови діаграм дуже велика, проте через те що додаток охоплює одразу багато сфер застосування вузькоспеціалізовані функції в ньому аж ніяк не реалізовані і нема стороніх модулів для їх реалізації. В Visual Paradigm неможливо провести жодних аналітичних функцій над побудованими схемами.

1.3. Сфери застосування

Розроблений веб-додаток навідміну від розглянутих вище може бути використаний у технічних сферах, оскільки зазначені вище застосунки не мають функціоналу, який спростив би користувачам проектування комбінаційних схем шляхом аналізу і виявлення помилок проектування.

Можливість мінімізувати задані схеми дозволить створювати мікросхеми більшої щільності й відповідно дешевщі. Данний функціонал дозволить зменшити кількість елементів на виготовлення кожного пристрою. Це стане у нагоді організаціям, що займаються розробкою радіо-пристроїв, обчислювальної техніки та майже усіх інших сфер апаратної розробки.

Крім того, завдяки побудові архітектури на основі абстрактних моделей, з’являються можливості для гнучкої кастомізації, такої як наприклад надає додаток Visual paradigm. Це довзолить за невелеких зусиль створювати системи аналізу прийнятя рішень та перегляду наслідків, або різноманітні діграми чи навіть моделі комп’ютерних мережей.

2. МЕТОДИ ТА ТЕХНОЛОГІЇ

2.1. Алгоритм подання логічних схем

Як зазначено вище існуючі програмні продукти у своїй більшості не можуть надати користувачу змогу створення власних логічних елементів і, відповідно, обмежені у можливості користувацької модифікації елементів.

В своїй дипломній роботі я вирішив цю проблему за допомогою створення функціоналу для задавання режиму роботи елементу у вигляді таблиці істиності.

Для редагування взаємозв’язків між елементами доступні програми мають однотипний підхід до редагування данних, шляхом перетягування елементів з панелі керування на схему і поєднування виходів та входів між ними за допомогою ліній. Про те такий інтерфейс не може коректно працювати у мобільному телефоні. У якості рішення було створено CRUD редактор схем, елементів та взаємозв'язків між портами елементів, що дало змогу користуватися додатком на будь-якій платформі. Оскільки у якості платформи для програмування було використано .Net Core отримана аплікація може бути запущена на будь-якій настільній операційній системі – Windows, Mac, Linux.

2.2. Алгоритм перевірки на мінімальність

На основі функцій алгебри логіки здійснюється побудова різних комбінаційних схем. Для створення найбільш простої (за кількістю використовуваних логічних елементів) схемної реалізації виконують перетворення функцій алгебри логіки. Значні труднощі виникають при вирішені цього завдання через необхідністю підтримки особливостей використовуваної елементної бази. Для вирішення цих труднощей були

винайдені методи, що дозволяють спрошувати функції булевої алгебри до виду, який містить мінімальну кількість термів.

Під мінімізацією логічної функції розуміється виконання перетворень з метою отримання найбільш простого уявлення функцій алгебри логіки. В інженерній практиці використовуються такі основні методи мінімізації:

- послідовного перебору
- послідовного спрощення аналітичного виразу
- карт Карно
- Квайна
- Квайна - Мак-Класки
- Л.Т. Мавренкова

Метод послідовного перебору заснований на простому переборі всіх можливих варіантів алгебраїчного виразу функції булевої алгебри. Використання методу не потребує спеціальних знань, проте, він застосовується лише для дуже простих функцій алгебри логіки, утворених від однієї або двох змінних. Данний метод через невиправдану кількість ресурсів для реалізації відразу відкинемо.

Метод послідовного спрощення аналітичного вираження базується на перетворенні функції алгебри логіки з використанням основних законів і тотожностей булевої алгебри. Перевагою методу є можливість його застосування для мінімізації будь-яких функцій алгебри логіки, представлених у вигляді аналітичного виразу. Однак розглянутий метод досить трудомісткий для виконання вручну, мало наочний та вимагає великого досвіду, уваги і інтуїції і, отже, легко призводить до виникнення помилок.

Карти Карно - це метод спрощення булевих виразів алгебри логіки. Карта Карно зменшує потребу в великих обчисленнях, оскікльки використовує можливості аналітичних функцій людини та їх здібності до розпізнавання образів через що є дуже наочним. Метод, заснований на застосуванні карт

Карно, передбачає завдання функцій алгебри логіки у вигляді координатних карт станів. Після запису функції в карту Карно відразу можна записати мінімальну форму функції, що істотно зменшує ймовірність появи помилки.

Метод Квайна заснований на послідовному застосуванні до ДСНФ операцій неповного склеювання і елементарного поглинання. Метод чудово підходить в основному для функцій алгебри логіки від невеликої кількості аргументів.

Метод Квайна-Мак-Класкі в своїй основі використовує запропонований Мак-Класкі зручний підход до практичної реалізації алгоритму Квайна. Під час використання даного методу функція алгебри логіки представляється однією з досконалих нормальних форм. Проте відміність від метода Квайна полягає в тому, що констітуенти одиниці та констітуенти нуля заданої функції записуються у вигляді умовних чисел, які називаються номерами відповідних констітуєнт, а не в натуральному вигляді. Такий підхід суттєво спрощує обробку великої кількості аргументів і значно зменшує ймовірність появи помилки на етапі виконання перетворень.

Як і метод Квайна-Мак-Класкі – метод Л.Т. Мавренкова базується на використаны умовних чисел, якы відповідають різним наборам значень аргументів для уявлення констітуєнт одиниц. Наочне відображення методу полягає у подані зв’язків між двійковими сусідніми з використанням спеціальної решітки. За допомогою операції склеювання робочих наборів, розташованих вздовж горизонтальних та вертикальних ліній решітки, відбувається мінімізація даної функції. [8]

Як один із варіантів можна розглянути метод невизначених коефіцієнтів. Метод невизначених коефіцієнтів є найбільш трудомістким з усіх представлених. Для мінімізації функції за допомогою цього методу бажано мати повну таблицю наборів чисел в ДСС.

Знаходження мінімальної ДНФ зводиться до вирішення системи лінійних рівнянь і складається з наступних етапів:

- Складаються рівняння з усіх наборів коефіцієнтів від 1 до N змінних.
- Викидаються рівняння, в яких значення функції дорівнює нулю, а також нульові коефіцієнти
- Виконується пошук найбільш часто зустрічаються терм мінімального рангу.
- Видаляються повторювані коефіцієнти.

Розглянутий метод невизначених коефіцієнтів ефективний, якщо число аргументів функції не більше, ніж 5 - 6. Це пов'язано з тим, що число рівнянь дорівнює 2 в ступені N. Більш ефективним є виписування не всіх можливих кон'юнкція для функції, а тільки тих, які можуть бути присутніми в ДНФ даної функції. На цьому заснований метод Квайна. При цьому передбачається, що функція задана у вигляді СДНФ. В даному методі елементарні кон'юнкції рангу N, що входять до ДНФ, називаються мінітермами рангу N.

Оскільки методи послідовного аналітичного спрощення та послідовного перебору дуже ресурсомісткі і важкореалізуємі в вигляді програмного коду розглянемо детальніше реалізацію карт карно та методу Квайна-Мак-Класкі.

Використання карт Карно дозволяє виконвувати мінімізацію логічної функції в процесі її складання. Для мінімізації логічного виразу в карті Карно виділяються всі контури містять по 2 в ступені N одиниць. Якщо який-небудь контур перетинає зміни значення однієї або декількох змінних, то ці змінні не вносяться в логічну функцію для даного контуру.

Карта заповнюється одиницями, кількість яких дорівнює кількості доданків в СДНФ, кожний доданок - 1 одиниця.

Для карт Карно існують правила мінімізації:

- Необхідно покрити контурами усі групи одиниць (ДНФ) у карті Карно. В рамках кожного контуру повинні знаходиться лише

одиниці – це відповідає операції склеювання - знаходження імплікант даної функції.

- Кількість одиниць контуру має бути ступенем двійки (1, 2, 4, 8, ...).
- Кожен контур буде відповідати простий імпліканті лише якщо він буде покривати максимально можливу кількість клітинок.
- Контури повині включати всі одиниці в карті (навіть поодинокі).
- Будь-яка одиниця може включатися безмежну кількість разів в контури.
- Безліч контурів, що покривають всі одиниці функції, утворюють кінцеву ДНФ.

Етапи мінімізації:

- Перехід від ДНФ до досконалої ДНФ. Членами будуть несклеювані елементарні кон’юнкції.
- Перехід до кінцевої формі.
- Перехід до мінімальної ДНФ.

На карті Карно мінімізація в своїй основі тримає ідею вибору контурів таким чином, щоб контури не перетинали кордон зміни якомога більшої кількості змінних. При дотримані цієї умови, кінцева функція вийде максимально мінімальної.

Як ми бачимо метод карт Карно є зручним методом мінімізації булевих функцій до 5 змінних. Але важко спростити булеві функції, що мають більше 5 змінних, використовуючи цей метод.

Метод Квайна-Мак-Класкі є табличним методом, заснованим на концепції простих імплікантів. Ми знаємо, що простий імплікант - не може бути додатково зменшений шляхом об'єднання з будь-яким іншим з членів даної булевої функції. Цей табличний метод корисний для отримання простих імплікантів шляхом повторного використання булевої операції поглинення.

Для спрощення булевих функцій за допомогою табличного методу Квайна-Мак-Класкі необхідно виконати наступні кроки:

Впорядковати задані терми в порядку зростання і створити групи на основі кількості присутніх одиниць у їх двійкових поданнях. Таким чином, отримуємо, щонайменше N+1 груп, при умові що є N булевих змінних у булевій функції або, як у випадку програмування алгоритму, N біт у двійковому еквіваленті мінімальних термінів.

Порівняти мінімальні терми, присутні в наступних групах. Якщо відбувається зміна тільки в одній бітовій позиції, то візьмемо пару з цих двох мінімальних терм. На отриманій позиції вионати поглинення, а інші біти залишити в тому ж вигляді.

Далі повторюємо попередній крок з новоутвореними термами, доки не буде отримано всіх простих імплікантів.

Сформулювати таблицю простих імплікант. Вона буде складатися з набору рядків і стовпців. Основні імпліканти можуть бути розміщені в рядку, а терми можуть бути поміщені в колонку. Помістити «1» у клітинки, що відповідають мінімальним термам, які покриваються кожним простим імплікатом.

Знайти прості імпліканти, переглянувши кожний стовбчик. Якщо мінімальний терм покривається тільки одним простим імплікантом, то він є кінцевим простим імплікантом. Ці незамінні імпліканти будуть частиною спрощеної булевої функції.

Зменшити таблицю основних імплікантів, видаливши рядок кожного основного первинного імпліканта і стовпці, що відповідають мінімальним термінам, які охоплюються цим основним імплікатором. Повторити попередній кро для таблиці зменшених основних імплікант. Зупинити цей процес, коли всі терми виконання цієї булевої функції закінчаться.

МОЖНО ЕЩЕ ПРО МЕТОД КВАЙНА-МАК-КЛАСКИ И МЕТОД КВАЙНА

2.3. Опис програмної платформи

Для реалізації програмного продукту в якості платформи було обрано саме веб-додаток, оскільки станом на 2019 рік кількість користувачів мережі інтернет, а відповідно і людей, що мають браузер значно перевищує кількість існуючих настільних комп'ютерів за рахунок мобільних платформ. Навіть на великих підприємствах не завжди є необхідність мати комп’ютер, а вистачає планшетів та смартфонів. Реалізація веб-додатку допоможе користувачеві аналізувати логічні схеми прямо з телефона під час знаходження біля їх матеріалізованого представлення, що дозволить зменшити витрати на транспортування обладнання, та збільшить продуктивність у виконані цих задач.

2.3.1. Користувацька частина

Після вибору платформи вибір технологій для реалізації користувацького інтерфейсу значно обмежився. Був проведен аналіз між двума мовами TypeSript та JavaScript. Нижче розглянемо переваги тієї та іншої технології:

TypeScript навідміну від JavaScript є строго типізованою компілюємою мовою програмування, через що є дуже важкою (в сенсі розміру). JavaScript інтерпретується бразуером, легший у вивчені та легковісніший.

JavaScript розроблений Mozilla, а TypeScript – Microsoft. Оскільки в якості мови програмування (про що описано нижче) використовується C#, що також створений Microsoft можно простежити сильний вплив цієї мови на структури TypeScript.

До сфер застосування TypeScript – можна віднести тільки клієнтську частину додатків, JavaScript в свою чергу можна використовувати також на серверній стороні.

Синтаксис TypeScript визначає набір правил написання, кожну специфікацію визначає її власний синтаксис. А програмний код складається з наступних складових:

- Модулі
- Функції
- Змінні
- Заяви та вирази
- Коментарі

Для виконання JavaScript в браузері необхідно лише обгорнути код тегами <script> //ваш код </script>, що дає браузеру інформацію, про те, що вміст цього тегу необхідно інтерпретувати.

Як результат можна виділити такі переваги TypeScript:

- Статична типізація
- Кращий вибір для великих проектів.
- Краще для спільної роботи. Якщо у великих проектах програмує багато розробників. У цей час існує ймовірність, що внесені зміни одним з розробників зможуть заважити іншому. Зростає кількість помилок, що ускладнює обробку. Таким чином, статична типізація довзоляє виявити помилки під час написання коду. Що робить збільшує продуктивність, а також дозволяє легше проводити відлагодження коду.
- Краща продуктивність. Особливості, такі як - о код ECMAScript 6, динамічне введення, автозаповнення допомагають розробникам підвищити продуктивність.

До переваг JavaScript можна віднести:

- Він має величезну активну спільноту розробників, яка робить його більш популярною мовою.

- Він інтерпретується одразу у браузерах. В той час, як для Typescript спочатку компілюється та перетворюється у JavaScript, а це ще один крок.
- Під час компіляції TypeScript в JavaScript створюється більше рядків коду ніж необхідно на сирому JavaScript, тому важче його підтримувати і шукати в ньому помилки (наприклад за допомогою Chrome DevTools).
- Більша гнучкість.

Оскільки Typescript - це об'єктно-орієнтована мова прогрмаування, то його доцільніше використовувати для великих проектів зі складним інтерфейсом користувача. Він дозволяє робити код більш узгодженим, чистим, простим і багаторазовим у використані в різних місьцях проекту. JavaScript краще використовувати у невеликих проектах, що як раз підходить для наших потреб.

Для реалізації користувацького інтерфейсу в якості основної мови програмування було використано JavaScript, оскільки ця мова дозволяє швидко візуалізувати будь-які контроли без особливих складностей та підтримується усіма сучасними браузерами.

Частина функціоналу базується на викликах API-методів серверної частини за допомогою технології AJAX. Зокрема завантаження сторінки перегляду схеми відбувається в два єтапи. На першому єтапі відмальовується меню та базова інформація про схему, а на другому відбуваються два API виклики для отримання таблиці істиності та данних, що до елементів схеми для їх подальшого відображення. Подання елементів схеми реалізовано на базі бібліотеки vis.js.

Vis.js - це динамічна бібліотека візуалізації на основі JavaScript. Бібліотека призначена для побудови графів, комп’ютерних мереж та діаграмм. Має широкий функціонал для обробки великих обсягів динамічних даних, а також для маніпулювання даними. Бібліотека розроблена Almende B.V. Vis.js

підтримує роботу в Chrome, Firefox, Opera, Safari, IE9 + і більшості мобільних браузерів (з повною підтримкою дотиків).

Для реалізації базових дій з DOM-деревом використано бібліотеку jQuery.

jQuery - це швидка, невелика та багатофункціональна бібліотека JavaScript. Вона дозволяє легко проходити по дереву елементів в рамках HTML та маніпулювати ними, обробляти події браузеру, створювати анімації, відправляти асинхроні запити до серверної частини і багато інших речей, крім того підтримується усіма сучасними браузерами. Завдяки поєднанню універсальності та розширюваності jQuery зробив велечезний вплив на підходи до програмування на JavaScript в цілому світі.

Для відображення данних було використано HTML (з деякою модифікацію у вигляді Razor cshtml, про що буде написано нижче) та CSS.

HTML - це мова для опису структури веб-сторінок. HTML дає авторам можливість:

- Публікувати онлайн-документи з заголовками, текстом, таблицями, списками, фотографіями тощо
- Отримувати онлайн-інформацію за допомогою гіпертекстових посилань одним натисненням кнопки.
- Розробкляти форм для проведення операцій з відправкою запитів, для використання в пошуку інформації, оформлення бронювання, замовлення продукції тощо.
- Включати розсилки, відеокліпи, звукові кліпи та інші медіафайли безпосередньо в веб сторінку.
- За допомогою HTML автори описують структуру сторінок за допомогою розмітки. Елементи мови позначають фрагменти змісту, такі як «абзац», «список», «таблиця» тощо.

CSS - це мова для опису стилів веб-сторінок. Вона дозволяє задавати кольори, макет і шрифти. Це дозволяє адаптувати зовнішній вигляд до різних типів пристроїв, таких як великі екрани, невеликі екрани або принтери. CSS не залежить від HTML і може використовуватися з будь-якою мовою розмітки на основі XML. Поділ HTML з CSS полегшує ведення сайтів, обмін таблицями стилів на сторінках і адаптування сторінок до різних середовищ. Це називається поділом структури (або: змісту) від подання. [5]

2.3.2. Серверна частина

Для створення серверної частини додатку були використані наступні технологій C#, .NET Core, MSSQL. Зупинимося на кожній детальніше.

C# - елегантна та строготипізована об'єктно-орієнтована мова, що дозволяє розробникам створювати різноманітні безпечні та надійні програми, які працюють на .NET Framework або .NET Core. Можна використовувати C# для створення клієнтських додатків Windows, Linux, MacOS, веб-служб, розподілених компонентів, клієнт-серверних додатків, додатків баз даних і інших програмних продуктів. Visual C# надає просунутий редактор коду, інтегрований відладчик та багато інших інструментів, які полегшують розробку додатків на основі мови C # і .NET Framework/.NET Core.

Синтаксис C # є дуже виразним, але він також простий і легкий в освоєнні. Синтаксис фігурної дужки C# буде миттєво впізнаваний всім, хто знайомий з C, C ++ або Java. Розробники, які знають будь-яку з цих мов, зазвичай здатні продуктивно працювати в C# протягом дуже короткого часу. Синтаксис C# спрощує багато складнощів C ++ і надає потужні функції, такі як типи значень з нулями, перерахування, делегати, лямбда-вирази і прямий доступ до пам'яті, які не може надати Java. C# підтримує загальні методи та типи, які забезпечують підвищену безпеку, ітератори, які дозволяють реалізовувати класи колекцій і визначати поведінку користувальницьких ітерацій, що потім можуть бути просто використанні клієнтським кодом.

Вирази з інтегрованими мовними запитами (LINQ) роблять строго типізований запит першокласною мовною конструкцією.[7]

Розробники, які ніколи не торкалися продуктів Microsoft, все ще вважають, що прийняття її інструментів створює необхідність у коштовних виділених серверах, обмеженість у виборі платформи однією Windows, та постійних нагадувань про продовження ліцензії, проблем з постійними оновлюваннями у самий незручний час. Але щоб бути справедливим, до недавнього часу це було схоже на правду. Пройде деякий час, перш ніж широка спільнота зможе позбутися цієї стигми. Це не цікаво, коли ви застрягли на машині для розробки під ОС Windows, і дорогі сервери Windows є вашим єдиним варіантом розгортання, тоді як світ набагато більший. Це була одна з причин, що заважала .NET і Microsoft завойовувати ринок. Важливо те, що з новою любов'ю Microsoft до відкритого коду ви можете бути впевнені, що C# має великі перспективи для подальшого розвитку.

Проте ваші турботи були б виправдані, якщо б Microsoft не прийняла відкритий код. Вона не змогла б конкурувати з тепер загальною ідеєю, що інструменти розробнки повинні бути відкритими, і існує багато альтернатив.

Після того, як Microsoft придбала GitHub, схоже, що Microsoft наближається до своїх коренів: створення інструментів розробки програмного забезпечення, що точно піде на користь розвитку C#.

Microsoft має багато років досвіду у розробці мов програмування, але через останні події легко забути про внески компанії до розробки програмного забезпечення. Наявність такої потужної сили, що стоїть за C# та її екосистеми повинна тримати C# на актуальних позиціях ще довгі роки.

У опитуванні розробників StackOverflow 2018 C# посів 8 місце у списку найпопулярніших технологій. У 2017 році C# був 4-м. За 4 роки до цього C # також був четвертим. «Падіння» з 4-го по 8-е місце між 2017 і 2018 рр. Можна пояснити частково, оскільки StackOverflow додав html і css до списку найбільш популярних технологій у 2018 році, і їх важко перевершити. Проте,

у 2018 році 34,4% розробників програмного забезпечення мають C# у своєму арсеналі.

Використовуючи Xamarin і C#, ви можете створювати нативні мобільні програми на iOS та Android (і Windows Phone.). З 2016 року компанія Xamarin є частиною корпорації Майкрософт, яка викупила її.

.NET Core є кроссплатформеним, вільною і відкритою системою керованого програмного забезпечення для операційних систем. .NET Core працює на Windows, Mac і Linux без додаткових зусиль. Це означає, що ви можете розробляти .NET-додатки з C# на вашому комп'ютері Mac або Linux. Корпорація Майкрософт надала всі інструменти та sdk для плавного розвитку на обох платформах.

.NET Core і ASP.NET Core є останніми ітераціями фреймворків .NET і ASP.NET. Ядро C# + ASP.NET - це потужна комбінація, що дозволяє створювати будь-які додатки для будь-яких платформ швидко, якісно та гнучко, а відкритість цих фреймоврків дозволяє не піклуватись про підтримку і їх перспективи.

Раніше програмування на C# означало використання Visual Studio. Деякі розробники зневажають цю IDE, тепер намає необхідності використовувати її для розробки. Можна відмовитися від Visual Studio, і віддати перевагу текстовому редактору та консолі командного рядка. Visual Studio має видання для Mac, але ще не таке розвинене як версія для Windows.

2.3.3. Взаємодія серверної частини та користувацького інтерфейсу

В якості архітектурного рішення була обрана REST-архітектура передбачає застосування таких методів або типів запитів HTTP для взаємодії з сервером:

- GET
- POST
- PUT
- DELETE

Сервіс, що побудований на архітектурі REST, називається службою RESTful.

Веб-сервіси пройшли довгий шлях з моменту створення. У 2002 році веб-консорціум оприлюднив визначення веб-сервісів WSDL і SOAP. Це сформувало стандарт того, як реалізуються веб-служби. У 2004 році веб-консорціум також випустив визначення додаткового стандарту під назвою RESTful. За останні роки цей стандарт став досить популярним. І використовується багатьма популярними веб-сайтами по всьому світу, які включають Facebook і Twitter. Цей стандарт дозволяє легко інтегруватися із додатками, що його підтримують, крім того існують можливості автоматичної генерації коду обгорток для роботи з такими сервісами, а також інструменти автоматичного документування написаних сервісів.

REST - це спосіб доступу до ресурсів, які лежать в певному середовищі. Наприклад, у вас може бути сервер, на якому можуть розміщуватися важливі документи або зображення або відео. Все це є прикладом ресурсів. Якщо клієнт, скажімо, веб-браузер потребує будь-якого з цих ресурсів, він повинен надіслати запит серверу для доступу до цих ресурсів. Тепер REST визначає спосіб доступу до цих ресурсів.

Найчастіше REST-стиль особливо зручний при створенні будь-якого роду додатків для маніпулювання данними. Саме REST стиль дозволяє поєднати естетичність серверного коду та гнучкість користувацького інтерфейсу, крім того за рахунок шаблонізованості це дозволяє швидко та інтуітивно користуватись усіма можливостями системи.

Ключовими елементами реалізації REST є:

- Ресурси - першим ключовим елементом є сам ресурс. Створений веб-додаток на сервері має записи декількох комбінційних схем. Припустимо, що веб-додаток розгорнуто локально, тоді для того, щоб отримати доступ до ресурсу інформації про схему за допомогою REST, можна оформити команду

http://localhost/schema/1 - ця команда повідомляє веб-серверу про надання інформації про схему з ідентифікатором 1.

- Дієслова (методи) запиту - це опис того, що ви хочете зробити з ресурсом. Браузер видає дієслово GET, щоб отримати інформацію з кінцевої точки. Проте є багато інших доступних дієслів, включаючи такі методи, як POST, PUT і DELETE. Таким чином, у випадку з прикладом http://localhost/schema/1 веб-браузер фактично видає GET метод, оскільки він хоче отримати інформацію про схему 1.
- Заголовки запитів - це додаткові інструкції, надіслані разом із запитом. Вони можуть визначати тип необхідної відповіді або деталі авторизації.
- Тіло запиту - дані надсилаються з запитом. Дані зазвичай надсилаються в запиті, коли запит POST подається на веб-службу REST. При використані методу POST клієнт фактично повідомляє веб-службі, що він хоче додати ресурс на сервер. Отже, тіло запиту повино мати деталі ресурсу, який потрібно додати на сервер.
- Тіло відповіді - це основна частина відповіді. Таким чином, у нашому прикладі, якщо ми хотіли б запитувати веб-сервер через запит http://localhost/schema/1, веб-сервер може повернути сторінку з даними працівника в тілі відповіді.
- Коди статусу відповіді є загальними кодами, які повертаються разом з відповіддю веб-сервера. Прикладом є код 200, який, як правило, повертається, якщо немає помилки при поверненні відповіді клієнту.

При вирішені задачі створення, редагування, перегляду та видалення для кожної моделі було створено окремий контролер, що мав щонайменше 5

методів. 3 методи з дієсловом GET: перший з них не приймає жодних параметрів, а повертає список усіх записів, що знаходяться під владою даного контролера, інші два приймають 1 параметр – ідентифікатор запису і відрізняються адресою запиту (при додавані перед ідентифікатором “details/” будуть відображені деталі запису в залежності від моделі це можуть бути пов’язані записи з інших моделей, тощо). Метод POST приймає в тілі запиту об’єкт, структура якого відповідає моделі контролера, проводить його валідацію та створює запис в базі даних, метод PUT приймає на вхід окрім об’єкту в тілі запиту, ще й параметр в адресі звернення, що повідомляє про ідентифікатор моделі, яку необхідно змінити.

В рамках додатку в якості заголовків для завантаження сторфінок використовується text/html. Для API-команд application/json або application/xml.

Для взаємодії між клієнтом та сервером у веб-додатку було створено два модулі, що відрізняються за підходами. Перший - для шаблонізації інтерфейсу редагування данних системи. Серед доступних технологій вибір стояв між побудовою інтерфейсу на основі API-контролерів, використані ASP.NET сторінок або використані технології Razor WebPages. Через простоту підтримки коду да можливості використання серверного коду при створені веб сторінок було обрано Razor.

Razor - це синтаксисичний набір конструкцій програмування для вбудовування коду сервера в веб-сторінки.

Синтаксис Razor базується на рамках ASP.NET, частині Microsoft.NET Core, спеціально розробленому для створення веб-додатків.

Синтаксис Razor надає можливості ASP.NET, але використовує спрощений синтаксис, який легше вивчати, що в свою чергу позитивно впливає на продуктивність. Веб-сторінки Razor можна охарактеризувати як HTML-сторінки з двома типами контенту: HTML-вміст і код Razor.

Коли сервер читає сторінку, він спочатку запускає код Razor, перш ніж він надсилає HTML-сторінку до браузера. Код, який виконується на сервері, може виконувати завдання, які неможливо виконати в браузері, наприклад, доступ до бази даних сервера та будь-який інший C# код. Код сервера може створювати динамічний вміст HTML на льоту, перш ніж він буде надісланий браузеру. Зібраний з браузера HTML-код, створений кодом сервера, не відрізняється від статичного вмісту HTML.

Веб-сторінки ASP.NET з синтаксисом Razor мають спеціальне розширення cshtml (Razor, використовуючи C #).[6]

Ця технологія дозволяє досить швидко будувати складні динамічні HTML сторінки, що потім відображаються на клієнтській стороні. Завдяки Razor побудовано CRUD інтерфейс для створення та редагування усіх данних, якими маніпулює програма.

Другим модулем став API-контроллер для виконання аналітичних функцій зі схемами та виконання інших вузьконаправлених операцій, таких як отримання моделі подання схеми, обробка збереження координат, розрахунок таблиці істиності та інших.

У .NET Core API представляється, як окремий спосіб побудови програми, який спеціально заточений для роботи в стилі REST (Representation State Transfer або "передача стану вистави"). Visual Studio дозволяє генерувати шалони классів для створення API контролерів, що полегшує структуризацію та читаємість коду і дозволяє швидко та легко переходити одразу до написання логіки.

По своїй суті Web API представляє собою веб-службу, до якої можуть звертатися інші додатки. Причому ці програми можуть представляти будь-яку технологію і платформу - це можуть бути веб-додатки, мобільні або десктопні клієнти. В наслідок чого веб-застосунок можна масштабувати за допомогою мікросервісної архітектури і поєднувати з програмними модулями написанами на будь-яких мовах програмування та з використанням будь-яких технологій.

.Net Core підтримує роботу з веб-сокетами, що дозволяє відправляти дані з сервера клієнту без додаткових запитів.

3. ВЕБ-ДОДАТОК

3.1. Інструкція користувача

Розроблений програмний продукт зустрічає користувача інтуітивно зрозумілим інтерфейсом. Перше, що бачить користувач – це список доступних схем. В залежності від пристрою інтерфейс адаптується під розмір екрану. За допомогою позначки з трьома лініями клієнт може відкрити контекстне меню в мобільній версії. На версії для настільних комп'ютерів контекстне меню видно одразу (рис.14).

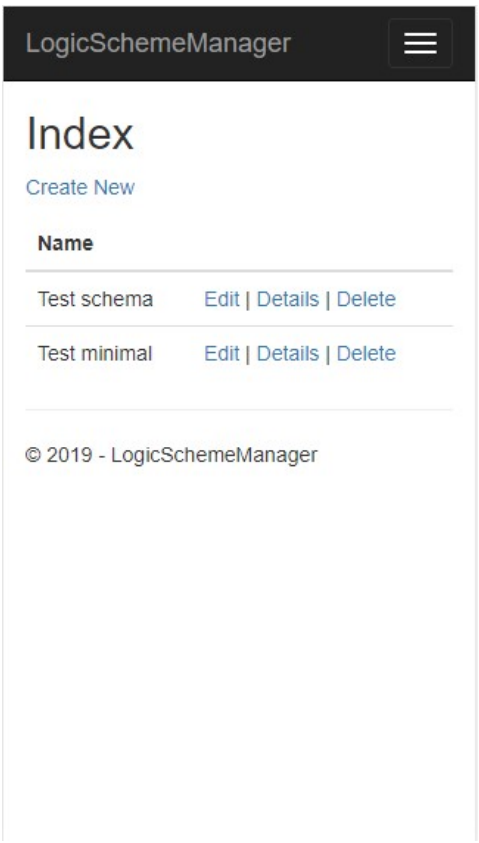


Рис. 8 - список схем

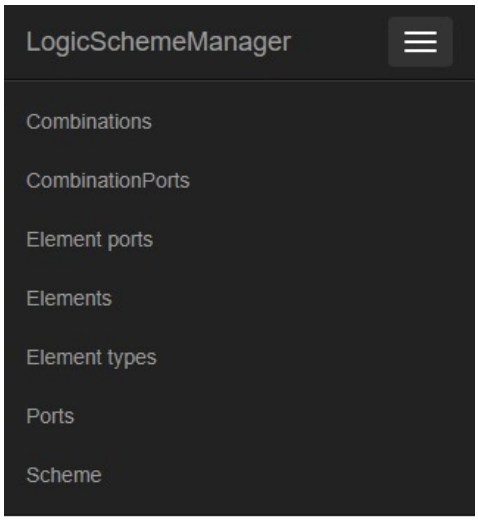


Рис. 9 контекстне меню

В контекстному меню користувач може одразу обрати розділ, що його цікавить. Кожний розділ створений для редагуваня окремої сутності, що відображена в базі даних.

Розглянемо розділи, необхідні для створення схем:

- Scheme – розділ, в якому знаходяться усі комбінаційні схеми створені користувачами.
- Elements – містить список усіх елементів, що створено в системі та їх зв'язки зі схемами. Саме тут користувач може створити додати необхідні елементи до своєї схеми і знайти помилку у випадку, якщо графічне подання схеми не буде відповідати схемі, яку користувач хотів внести в систему.(рис. 10)
- Element Ports – відображає інформацію про взаємозв'язки між елементами системи. Саме в цьому розділі набір елементів перетворюється в схему. При необхідності тут можно дуже швидко та легко скоректувати логіку схеми. (рис. 11)

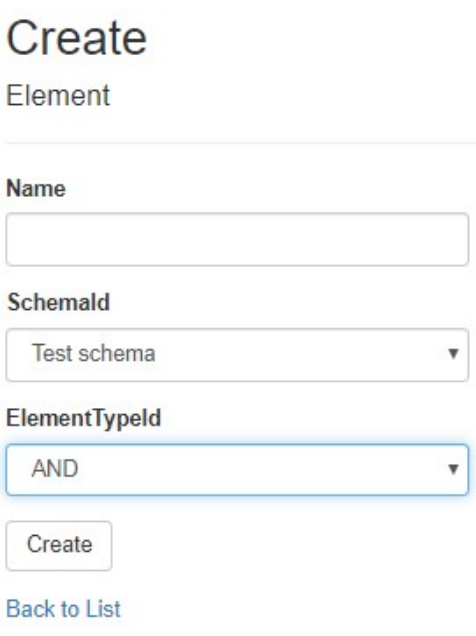


Рис. 10 створення елементу

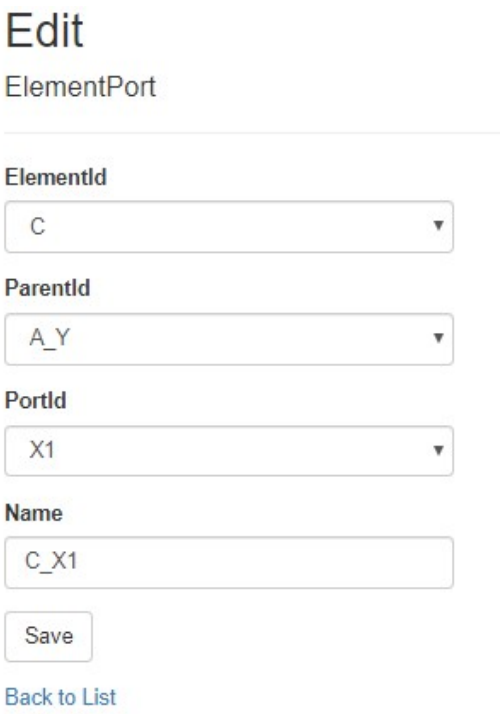


Рис. 11. створення зв'язку

Розглянемо розділи, що необхідні для створення нових типів елементів:

- Element Types – це розділ, призначений для створення нових типів елементів та редагування існуючих. Для створення нового типу

елементу користувачеві необхідно буде задати йому назву та внести для нього таблицю істиності.

- Combinations – призначений для створення таблиць істиності для типів елементів. В даному розділі можна задати ім’я будь якої комбінації для спрощення подальшого пошуку та використання її в системі. На рисунку 12 зображені комбінаціїї для елементів типу AND. Для наочності в їх назві відоражено вектор входів та значення виходу при заданому векторі.
- Combination Ports – зберігає в собі значення портів в рамках кожної комбінації. Цей розділ дозволяє користувачу знайти необхідне значеня таблиці істиности елементу та створювати нові значення для нових типів елементів. Переглянути значення портів для елементів з типом AND можна на рисунку 13.
- Ports – розділ, що необхідний для зберігання інформації про типи портів (вхід або вихід) та стандартні назви для їх подальшого використання при створені взаємозв'язків між елементами схеми. Для створення багатовходових елементів спочатку необхідно створити нові порти та дати їм назви у цьому розділі, після чого їх можна буде викоритовувати для створення векторів комбінацій.

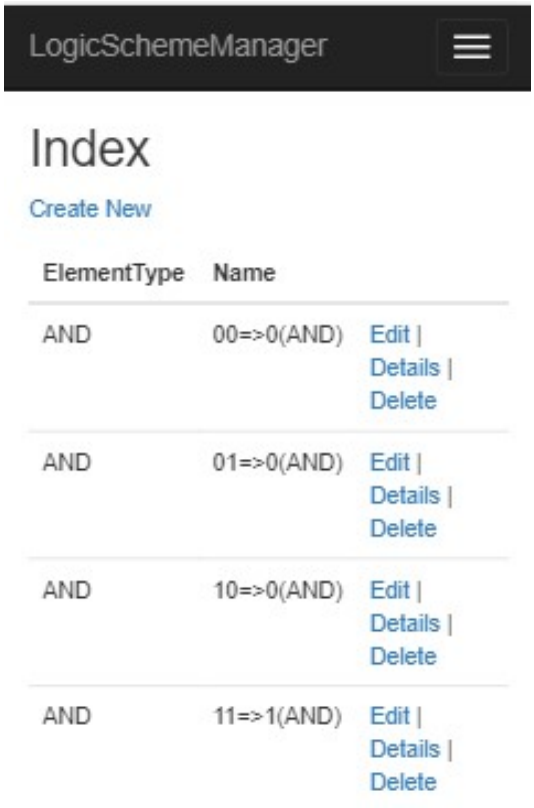


Рис. 12 розділ «Комбінації»

Combination	Port	Value
00=>0(AND)	X1	☐
00=>0(AND)	X2	☐
00=>0(AND)	Y	☐
01=>0(AND)	X1	☐
01=>0(AND)	X2	☑
01=>0(AND)	Y	☐
10=>0(AND)	X1	☑
10=>0(AND)	X2	☐
10=>0(AND)	Y	☐
11=>1(AND)	X1	☑
11=>1(AND)	X2	☑
11=>1(AND)	Y	☑

Рис 13 значеня портів комбінацій

Власне найцікавіше чекає користувача в деталях схеми. Для того, щоб туди потрапити достатньо в розділі “Scheme” нажати на гіперсилку “Details” біля потрібної схеми. Сторінка зустрічає користувача поданням схеми і графічному вигляді, як зображено на рисунку 14. Користувач може змінити розташування елементі та зберегти свої зміни натисканням на кнопку “Save coordinates”. При натискані на елемент його зв’язки з іншими елементами стають жирнішими, а обраний елемент підсвічується синім кольором. Це особливо актуально при перегляді великих схем. За допомогою перетягування можна змінити поле перегляду схеми, а при прокрутці колесика миші можно наблизити окремі участки схеми або віддалити схему для загального перегляду. На мобільному телефоні для перегляду схеми використовуються “touch” події, відоповідно для збільшення схеми необхідно розвести два пальці на екрані, або звести для зменшення масштабу перегляду. За допомогою перетягування можно посунути схему до необхідного елементу.

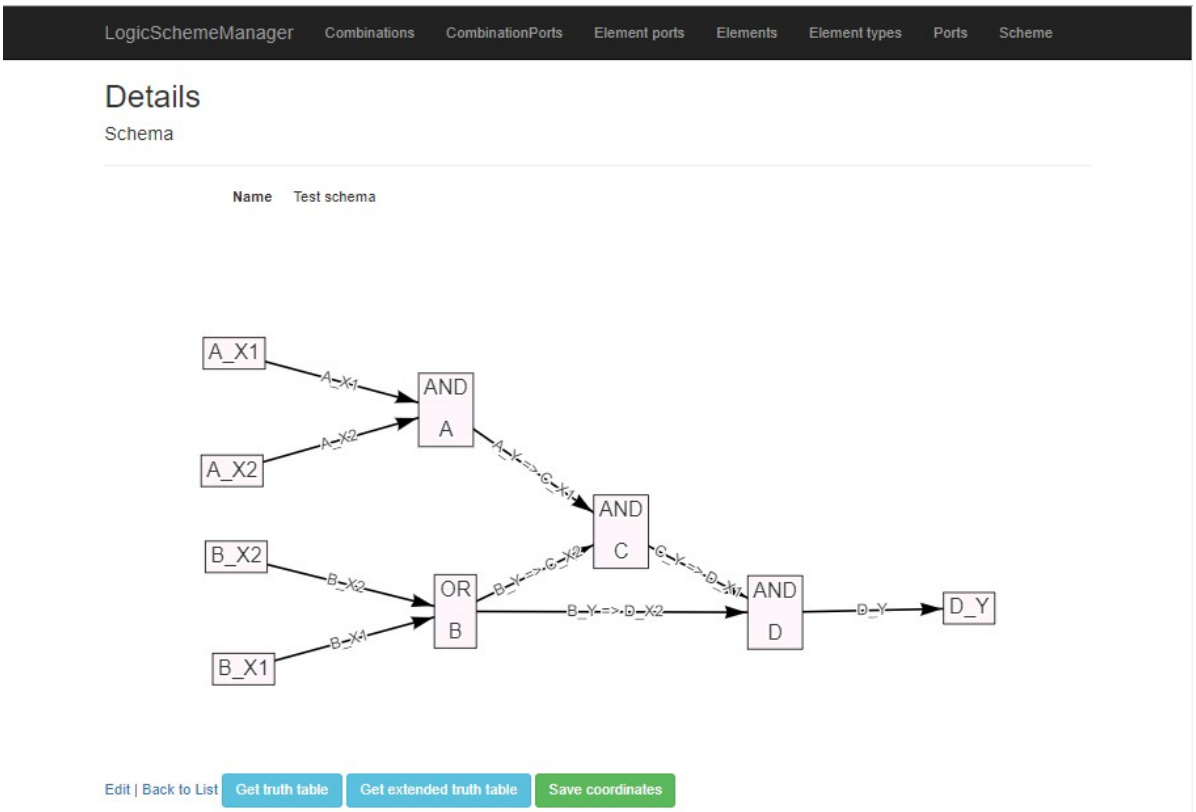


Рис 14 деталі комбінаційної схеми

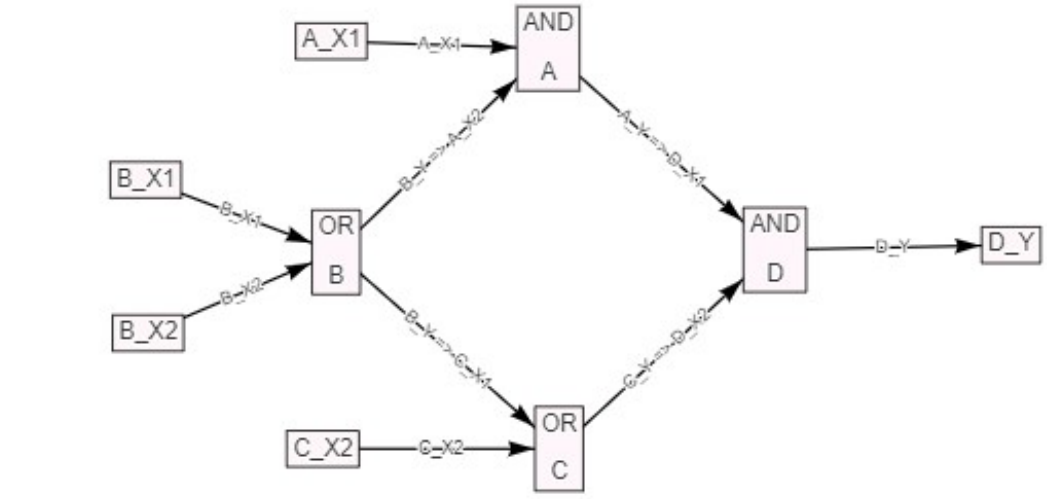
Для перегляду таблиці істиності створено дві окремі кнопки, кожна з яких відоповідає за різні варіанти деталізації данних. При натискані на кнопку “Get truth table” користувачу буде надано можливість переглянути звичну таблицю (рисунок 15), в той час як кнопка “Get extended truth table” додатково надасть інформацію про стан кожного з портів усіх елементів, що беруть участь у даній схемі (рисунок 16).

A_X1	A_X2	B_X1	B_X2	D_Y
0	0	0	0	0
1	0	0	0	0
0	1	0	0	0
1	1	0	0	0
0	0	1	0	0
1	0	1	0	0
0	1	1	0	0
1	1	1	0	1
0	0	0	1	0
1	0	0	1	0
0	1	0	1	0
1	1	0	1	1
0	0	1	1	0
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

A_X1	A_X2	B_X1	B_X2	C_X1	C_X2	D_X2	D_X1	A_Y	B_Y	C_Y	D_Y
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	1	0	0	0
0	0	1	0	0	1	1	0	0	1	0	0
1	0	1	0	0	1	1	0	0	1	0	0
0	1	1	0	0	1	1	0	0	1	0	0
1	1	1	0	1	1	1	1	1	1	1	1
0	0	0	1	0	1	1	0	0	1	0	0
1	0	0	1	0	1	1	0	0	1	0	0
0	1	0	1	0	1	1	0	0	1	0	0
1	1	0	1	1	1	1	1	1	1	1	1
0	0	1	1	0	1	1	0	0	1	0	0
1	0	1	1	0	1	1	0	0	1	0	0
0	1	1	1	0	1	1	0	0	1	0	0
1	1	1	1	1	1	1	1	1	1	1	1

Рис. 15 таблиця істиності Рис. 16 розгширена таблиця істиності

Для перевірки мінімальності необхідно виконати мінімізацію функції. Для запуску мінімізації необхіжно натиснути кнопку “Minimize”. Дана кнопка стане активной лише після побудови таблиці істиності. Приклад мінімізації можно побачити на рисунку 17. Як ми бачимо з прикладу результат мінімізації дозволив скороти кількість входів і виявив, що вхід A_X2 є надлишковим.



$$y = (x_{b\_x1}\, x_{a\_x1}) \vee (x_{b\_x2}\, x_{a\_x1})$$

Edit | Back to List | Get truth table | Get extended truth table | Minimize | Save coordinates

© 2019 - LogicSchemeManager

Рис. 17 Приклад мінімізації

3.2. Опис розроблених модулів

Для реалізації функціоналу зазначеного вище було створено наступні модулі:

- Модуль маршрутизації відповідає за маршрутизацію користувацьких запитів в середені системи та їх направлення до необхідного контролера для подальшої обробки.
- Модуль типізації даних відповідає за створення векторів для таблиць істиності з результатів компіляції схеми, а також за перетворення схеми у булеву функцію та перетворення елементів

та їх взаємозв’язків в обидві сторони для графічного подання та збережння данних про положення елементів в схемі з користувацького інтерфейсу відповідно.

- Модуль міжсерверної взаємодії представлений у вигляді API-контролерів та необхідний для інтегрування в стороні системи та посилання асинхроних запитів з клієнтського коду.
- Модуль компіляції схем відовідає за створення данних про поведінку схеми за різних умов вхіднних даних, якими потім маніпулює модуль типізації даних і передає їх до модулю міжсерверної взаємодії або до модулю перегляду результатів аналізу схеми.
- Модуль розробки елементів надає користувачу змогу створювати власні елементи будь-якої кількості входів та виходів шляхом заданя їх таблиці істиності, а також типізувати назви залежно від сфери використання та необхідних стандартів.
- Модуль перегляду результатів аналізу схеми дає можливість спостерігати таблиці істиності та результати мінімізації схем.
- Модуль створення схем відповідає за побудову комбігаційних схем з елементів та їх взаємозв’язків.
- Модуль мінімізації схем створений для використання функцій алгебри логіки за для мінімізації функції.
- Модуль виконання функцій алгебри логіки містить методи для перевірки можливостей застосування тих чи інших операцій над операндами та їх безпосереднє виконання під час мінімізації.

3.3. Швидкодія та гнучкість модулів

За рахунок використання асинхроних запитів до серверної частини вдалося забезпечети плавну роботу інтерфейсу без перезавантаження сторінок. Обробка запитів так само відбувається асинхроно, це створено за для

збільшення швидкодії під час отиримання та обробки данних з бази даних. Для повернення результатів API-контролерами використовується Newtonsoft JSON, що є на даний момент найшвидшим рішенням для серіалізації даних в середовищі .NET Core. За обробку запитів до базиданих відповідає EntityFramework, цей фреймворк дозволяє писати запити за допомогою LINQ в об’єктному стилі, завдяки чому можна дуже швидко та легко кастомізувати систему під себе або дописати додаткові модулі не вдаваючись в деталі роботи з MSSQL.

Завдяки модулю маршрутизації всі запити дуже легко можна перехопити та обробити навіть не знаючи як реалізована інша логіка, відділена об’єктна модель дозволяє без перешкод її розширити користуючись успадкуванням. Наприклад для доробки логіки створення простих бізнес-процесів з читання даних буде достатньо створити ще один класс, прописати в ньому логіку обробки вхідного двійкового вектору, та додати новий тип елементу наприклад «читання данних», після чого можна буде додати API – метод, що буде приймати в аргументах вектор значень і запускати компіляцію схеми з цим вектором. Оскільки компілятор схем працює с кожним елементом окремо, можно пронаслідуватись від елементу, та перевизначити метод компіляції, у випадку, якщо тип елементу – «читання данних», і запустити обробку вхідного вектору для цього елементу. Навіть на основі існуючого інструментарію вже можна моделювати різні ситуації та шляхи розгортання подій. З невеликою доробкою можна змінювати графічне подання комбінційних схем наприклад на блок-схеми, чи навіть додати зображення для кожно елементу, а за рахунок можливості зміни назв елементів можна наглядно дивитись що відбудеться при збігу тих чи інший подій, зо може стати в нагоді при аналізі ризиків, проектувані алгоритмів, тошо.

Предствалення всіх елементів в системі представляє собою щонайменше три сторінки: додаваня, редагування та деталей. Саме сторінка деталей буде цікава для майбутнього розширення та спрощення взаємодії з користувачем,

оскільки на цю сторінку користувач може потрапити лише після створення об’єкту – ми можемо створити модуль управління зв’язаними сутностями, в чому нам допоможе завчасно спланова архітектура моделей, де кожна модель має списки пов’язаних сутностей кожного типу, крім того, для кожного такого зв'язку автоматично створюється зовнішній ключ в базі даних. Відповідно з кожної пов’язаної сутності є обратне посилання. Саме побудова зовнішніх ключів значно пришвидшує процес обробки запитів базою даних. Усі записи, що формуються через прошарок роботи з базою оброблюються формувачем запитів і параметризуються. Що дозволяє будувати плани запитів, які SQL Server потім може перевикористовувати, тим самим не будуючи їх кожен раз знову.

ВИСНОВКИ

Не дивлячись на можливості вдосконалення вже на даному етапі розроблений веб-додаток може бути повністю самостійним продуктом, що стане у нагоді при вивчені булевої алгебри або прикладної теорії цифрових автоматів, а також для проектування обчислювалювальних пристроїв.

Розроблений веб-додаток покриває значну частину спектру потреб користувачів при проектувані схем без пам’яті, дозволяючи створювати схеми будь-якої складності і проводити над ними аналіз, як візувальний, так і автоматичний (мінімізувати функцію, яку реалізує схема).

При розробці було зазделегіть заплановано можливості його вдосконалення та розширення. Більша частина логіки базується на абстрактних моделях, це дозволяє перевизначати їх поведінку в залежності від потреб користувача при внесені незначних змін в кодову базу проекту.

Серед основними векторів майбутнього розвинення можна виділити:

- Написання платформи користувацького налаштування послідовностей виконання операцій і умовних порівнянь.
- Створення інтерфейсу для інтеграції користувацьких модулів поглибленного аналізу комбінаційних схем.
- Реалізація підтримки різноманітних стандартів подання логічних схем.
- Моделювання роботи комп’ютерних систем, запуск потактового виконання проходження сигналу по схемі.
- Можливості виконання тестів швидкодії створених схем
- Створення складних схем шляхом комбінації існуючих.
- Додавання електрокомпонентів для моделювання пристроїв.
- Реалізація витравлення друкованих плат на основі створених схем.

- Створення фукціоналу автоматичної побудови мінімізованих схем та схем, заданих таблицею істиності.
- Створення шаблонів завдань для побудови комбінаційних схем для освітніх цілей.

Розроблений продукт буде викладано у публічний репозиторій, що довзволить іншим розробникам покращити його та створювати власні модифікації.

Окрім позитивних моментів було виявлену суттєву проблему з швидкодією мінімізації схем при великій кількості входів. Нажаль система не володіє людським досвідом і не вчиться на проведеному аналізі. В майбутньому можно створити функціонал, що дозволить швидше приймати рішення у випадку знаходження в пам'яті додатку подібних схем, або шляхом декомпозиції схем на окремі ділянки та їх часткового аналізу.

Також планується доробити поділ на ролі користувачів системи для колективної роботи в освітніх та комерційних цілях. Створення вспливаючих підказок дасть змогу підвищити чуйність інтерфейсу. Сбір даних про дії користувачів дозволить додати автозаповнювані поля на основі штучного інтелекту та модуль передбачення помилок.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Моделирование. Тестирование, надёжность, контроль и диагностика компьютерных систем [Електронний ресурс]: учебное пособие для изучения дисциплин «Моделирование» и «Тестирование, надёжность, контроль и диагностика компьютерных систем» для иностранных студентов специальности «Компьютерные системы и сети», «Системное программирование» и «Специализированные компьютерные системы» / НТУУ «КПІ» ; сост. В. В. Гроль, В. А. Романкевич, Е. Р. Потапова.– Електронні текстові дані (1 файл: 782.5 Кбайт). – Киев: НТУУ «КПИ», 2011

2. Гроль В.В., Романкевич В.О. Базові поняття і конструкції мови програмування Сі. Методичні вказівки до вивчення дисципліни “Моделювання” // Київ.– “Політехніка”, 2003.– 24с.

3. Самофалов К.Г., Романкевич А.М., Валуйский В.Н., Каневский Ю. С., Пиневич М.М. Прикладная теория цифровых автоматов. – К.: Вища Школа. – 1987г. – 375с.

4. Гроль В.В., Романкевич В.А., Потапова Е.Р. Организация процедур логического моделирования цифровых блоков. Методические указания к изучению дисциплин «Моделирование», «Тестирование, надёжность, контроль и диагностика компьютерных систем» // Київ.– “Принт-центр”, 2007.– 44с

5. HTML & CSS – W3C — 2016. [Електроний ресурс]. URL https://www.w3.org/standards/webdesign/htmlcss (дата звернення: 20.05.2019).

6. ASP.NET Razor - C# and VB Code Syntax — 2003 - 2019. [Електроний ресурс]. URL https://www.w3schools.com/asp/razor_syntax.asp (дата звернення: 21.05.2019).

7. Офіційний сайт документації до продукції компанії Microsoft [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Редмонд – Режим доступу:

https://docs.microsoft.com (дата звернення 24.05.2019) – Technical documentation, API, and code examples

8. Національна бібліотека им. Н. Е. Баумана [Електроний ресурс]. URL https://ru.bmstu.wiki/Методы_минимизации_функций_алгебры_логики (дата звернення 19.05.2019) Методы минимизации функций алгебры логики

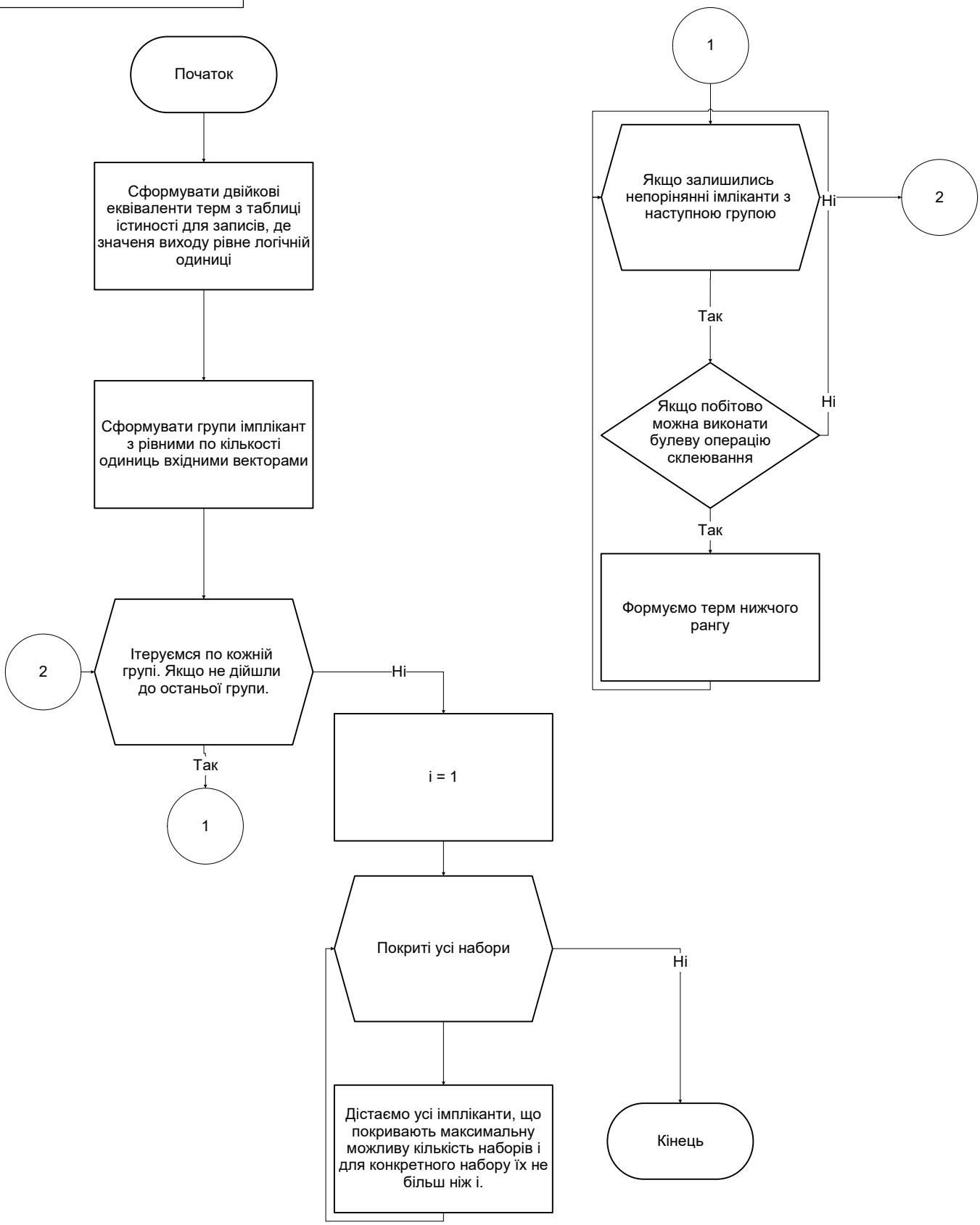
ІАЛЦ.045440.005 Д1

					ІАЛЦ.045440.005 Д1
Зм.	Арк.	№ докум.	Підпис	Дата	
Розроб.		Марченко О.Б.			База даних. Схема структурна.
Перев.		Потапова К.Р.			
Т.контр.					
Н.контр.		Клятченко Я.М.			
Затв.		Тарасенко В.П.			

Літ.	Маса	Масштаб
Аркуш 1	Аркушів 1	

КПІ ім. Ігоря Сікорського, ФПМ, КВ-51

					ІАЛЦ.045440.006 Д2		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Марченко О.Б.			Компіляція схеми. Схема Алгоритму	Літ.	Маса / Масштаб
Перев.		Потапова К.Р.				Аркуш 1	Аркушів 1
						КПІ ім. Ігоря Сікорського, ФПМ, КВ-51	
Т.контр.							
Н.контр.		Клятченко Я.М.					
Затв.		Тарасенко В.П.					



Зм.	Арк.	№ докум.	Підпис	Дата
Розроб.		Марченко О.Б.		
Перев.		Потапова К.Р.		
Т.контр.				
Н.контр.		Клятченко Я.М.		
Затв.		Тарасенко В.П.		

ІАЛЦ.045440.007 Д3

Мінімізація функцій.
Схема Алгоритму

Літ.	Маса	Масштаб
Аркуш 1	Аркушів 1	

КПІ ім. Ігоря Сікорського, ФПМ, КВ-51

					ІАЛЦ.045440.008 Д4	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розроб.		Марченко О.Б.			Міжмодульна взаємодія. Схема структурна	Літ. / Маса / Масштаб
Перев.		Потапова К.Р.				Аркуш 1 / Аркушів 1
						КПІ ім. Ігоря Сікорського, ФПМ, КВ-51
Т.контр.						
Н.контр.		Клятченко Я.М.				
Затв.		Тарасенко В.П.				